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

RE-IDENTIFIKACE VOZIDEL POMOCÍ VISION TRANSFORMERŮ

VEHICLE RE-IDENTIFICATION USING VISION TRANSFORMERS

DIPLOMOVÁ PRÁCE
MASTER'S THESIS

AUTOR PRÁCE **Bc. ZDENĚK JELÍNEK**
AUTHOR

VEDOUCÍ PRÁCE **Ing. JAKUB ŠPAŇHEL**
SUPERVISOR

BRNO 2023

VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
FAKULTA INFORMAČNÍCH TECHNOLOGIÍ

Zadání diplomové práce

144763

Ústav:	Ústav počítačové grafiky a multimédií (UPGM)
Student:	**Jelínek Zdeněk, Bc.**
Program:	Informační technologie a umělá inteligence
Specializace:	Počítačové vidění
Název:	**Re-identifikace vozidel pomocí vision transformerů**
Kategorie:	Zpracování obrazu
Akademický rok:	2022/23

Zadání:

1. Seznamte se s principy re-identifikace objektů a vyhledávání obrazů na základě vizuálních podobností.
2. Prostudujte dostupné materiály na dané téma, zaměřte se na metody využívající vision transformery.
3. Obstarejte si vhodnou datovou sadu pro re-identifikaci vozidel.
4. Vyberte vhodné metody založené na konvolučních neuronových sítí pro re-identifikaci objektů / vyhledávání obrazů na základě podobností *(image retrieval / metric learning)* a experimentujte s nimi.
5. Proveďte experimenty s vybranými metodami a diskutujte dosažené výsledky.
6. Vytvořte plakát a video prezentující vaší práci, její cíle a výsledky.

Literatura:

- Yang, Xipeng, et al. "Box-grained reranking matching for multi-camera multi-target tracking." *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2022.
- Liu, Chong, et al. "City-scale multi-camera vehicle tracking guided by crossroad zones." *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2021.
- Zhong, Yaoyao, and Weihong Deng. "Face transformer for recognition." *arXiv preprint arXiv:2103.14803* (2021).
- Dále dle pokynů vedoucího

Při obhajobě semestrální části projektu je požadováno:

- Splnění prvních tří bodů zadání.
- Značně rozpracovaný čtvrtý bod zadání.

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

Vedoucí práce:	**Špaňhel Jakub, Ing.**
Vedoucí ústavu:	Černocký Jan, prof. Dr. Ing.
Datum zadání:	1.11.2022
Termín pro odevzdání:	17.5.2023
Datum schválení:	31.10.2022

Abstrakt

Hlavním cílem této práce bylo zjištění možností vision transformerů při re-identifikaci vozidel. V této oblasti počítačového vidění doposud dominují konvoluční neuronové sítě. Celkem byly vyzkoušeny dva modely – TransReID a CMT. TransReID je model založený čistě na vision transformerech a byl vytvořený přímo pro re-identifikaci vozidel. Hlavní část experimentů s tímto modelem jsem věnoval využití klíčových bodů na vozidle. Při správné extrakci oblastí okolo klíčových bodů a využití postprocessingu jsem dosáhl state-of-the-art výsledků. Model CMT je kombinací konvolučních sítí a transformerů, který nebyl vytvořen pro re-identifikaci vozidel. Model jsem upravil a provedl s ním rozsáhlé experimenty pro získání nejlepší konfigurace pro re-identifikaci vozidel. Modely jsem vyhodnotil na standardních datasetech VeRi-776, VehicleID, CityFlowV2-ReID a CarsReId74k a porovnal se state-of-the-art modely. S modelem CMT jsem dosáhl na datasetu VeRi-776 nejlepšího výsledku 0,860 na metrice mAP a na datasetu VehicleID jsem dosáhl nejlepšího výsledku 97,6% na metrice Rank5.

Abstract

The main objective of this thesis was to investigate the potential of vision transformers in vehicle re-identification. Convolutional neural networks have so far dominated this area of computer vision. In total, two models have been tested – TransReID and CMT. TransReID is a model based purely on vision transformers and was created specifically for vehicle re-identification. The main part of the experiments with this model was devoted to the use of key points on the vehicle. With proper extraction of the regions around the key points and the use of post-processing, I achieved state-of-the-art results. The CMT model is a combination of convolutional networks and transformers that was not designed for vehicle re-identification. I modified the model and conducted extensive experiments with it to obtain the best configuration for vehicle re-identification. I evaluated the models on the standard datasets VeRi-776, VehicleID, CityFlowV2-ReID and CarsReId74k and compared with state-of-the-art models. With the CMT model, I achieved the best result of 0.860 on the mAP metric on the VeRi-776 dataset and the best result of 97.6% on the Rank5 metric on the VehicleID dataset.

Klíčová slova

vision transformer, re-identifikace vozidel, TransReID, CMT, VeRi-776, CityFlowV2-ReID, VehicleID, CarsReId74k

Keywords

vision transformer, vehicle re-identification, TransReID, CMT, VeRi-776, CityFlowV2-ReID, VehicleID, CarsReId74k

Citace

JELÍNEK, Zdeněk. *Re-identifikace vozidel pomocí vision transformerů.* Brno, 2023. Diplomová práce. Vysoké učení technické v Brně, Fakulta informačních technologií. Vedoucí práce Ing. Jakub Špaňhel

Re-identifikace vozidel pomocí vision transformerů

Prohlášení

Prohlašuji, že jsem tuto diplomovou práci vypracoval samostatně pod vedením pana Ing. Jakuba Špaňhela. Uvedl jsem všechny literární prameny, publikace a další zdroje, ze kterých jsem čerpal.

.....
Zdeněk Jelínek
14. května 2023

Poděkování

Tímto bych chtěl poděkovat panu Ing. Jakubovi Špaňhelovi za jeho čas, cenné rady, vypůjčení vybavení a dohled při psaní této práce.

Obsah

Seznam obrázků

Kapitola 1

Úvod

Ve své diplomové práci se zabývám re-identifikací vozidel, což je úloha počítačového vidění, která se zaměřuje na nalezení stejné identity vozidla na různých snímcích z různých kamer. Během let vznikla celá řada postupů a metod, jak řešit re-identifikaci vozidel. Nejrozšířenější přístupy využívají konvoluční neuronové sítě, které dosahují state-of-the-art výsledků. V této práci jsem se zaměřil na metody, které jsou založené na modelech vision transformerů. Vision transformery vycházejí z architektury modelu transformer, který je nejvíce využíván pro zpracování přirozeného jazyka. Vision transformery byly úspěšně využity v řadě úloh počítačového vidění, jako jsou klasifikace obrazů, segmentace obrazů či detekce objektů. V oblasti re-identifikace vozidel nejsou vision transformery natolik prozkoumány, jako je tomu např. u konvolučních sítí.

Cílem této práce bylo důkladné otestování a zjištění možností vision transformerů při re-identifikaci vozidel a získání porovnání s konvolučními sítěmi.

Kapitola 2 se věnuje popisu teorie transformerů a vision transformerů, která je potřeba znát pro jejich pochopení a úspěšnou implementaci. Závěrem kapitoly je popis triplet loss pro trénování modelů a metoda rerankingu.

V následující kapitole 3 detailně popisuji přístupy k re-identifikaci vozidel a jejich vývoj. Dále popisuji vybrané modely založené na vision transformerech, se kterými později experimentuji.

V kapitole 4 probírám datové sady vhodné k re-identifikaci vozidel a jejich porovnání. Součástí této kapitoly jsou metody datové augmentace.

Provedené experimenty s vybranými modely TransReID a CMT jsou popsané v kapitole 5. Součástí této kapitoly jsou také evaluační metriky. Detailně popisuji vývoj testování s uvedenými modely a jejich postupné zlepšování.

Závěrečná kapitola 6 rekapituluje dosažené výsledky, navrhuje budoucí vývoj a uzavírá tuto práci.

Kapitola 2

Pokročilé architektury neuronových sítí

V rámci své práce se zaměřuji na problematiku re-identifikace vozidel s využitím vision transformerů. Tyto moderní modely neuronových sítí jsou inspirované architekturou transformerů, která se používá zejména pro zpracování přirozeného jazyka. V této kapitole popisuji teorii, která stojí za vision transformery. Konkrétně se věnuji popisu teorie k transformerům obecně a poté se zaměřuji na teorii k vision transformerům.

2.1 Transformery

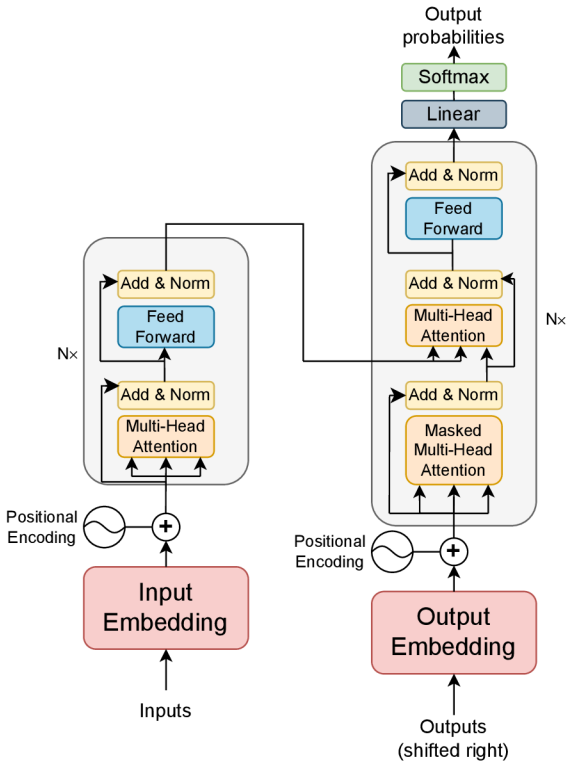
Transformery byly poprvé představeny v práci od *Vaswani et al.* s názvem *Attention Is All You Need* [29]. V této práci autoři navrhli model, jehož architektura nevyužívá rekurenci ani konvoluci a místo toho je založena na tzv. *attention* mechanismu, který umožňuje modelu zaměřit se na důležité části vstupních dat a ignorovat méně významné informace. Tento model se používá zejména pro úlohy zpracování přirozeného jazyka, jako je strojový překlad, rozpoznávání řeči nebo generování textu. Následující obsah této kapitoly vychází z jejich práce.

Navržený transformer má strukturu *kodér-dekodér*. Vstupní data (tokeny) jsou nejprve zpracována embedding vrstvou, která vstupní tokeny převede na číselné vektory, zvané *embeddingy*, o rozměru d. Tyto embeddingy se pak předávají do kodéru, který provede několik transformací, tzv. *self-attention*, *multi-head attention* a *feed-forward* (detailněji popsané dále v kapitole), a na výstupu poskytne nový embedding. Dekodér pak pro vygenerování výstupního symbolu využívá výstupní embedding kodéru spolu s výstupy dekodéru zpracované v předchozích krocích (autoregrese).

Celková architektura transformeru je zobrazena na obrázku 2.1.

Kodér

Kodér je složen z N identických vrstev, kde každá vrstva obsahuje dvě podvrstvy. První podvrstvou je *multi-head self-attention* vrstva, která umožňuje modelu soustředit se na významné části vstupní sekvence a přiřadit jim váhu v závislosti na kontextu. Druhou podvrstvou je plně propojená dopředná vrstva, která zpracovává výstup z první podvrstvy a vytváří nové reprezentace pro další zpracování. Po každé podvrstvě je provedeno reziduální propojení, kdy se výsledky z předchozí vrstvy přičtou k výsledkům aktuální vrstvy. Na závěr následuje normalizace vrstvy.



Obrázek 2.1: Architektura modelu transformer. Skládá se ze dvou bloků, kodér (vlevo) a dekodér (vpravo). Oba bloky využívají multi-head attention bloky. Obrázek převzat z [29].

Výstupem každé podvrstvy je tedy $LayerNorm(x + Sublayer(x))$, kde $Sublayer(x)$ představuje implementaci příslušné podvrstvy a $LayerNorm$ představuje normalizaci. Výstup poslední vrstvy kodéru se používá jako vstup do dekodéru pro další zpracování a generování výstupu.

Dekodér

Dekodér je také složen z N identických vrstev. Oproti kodéru obsahuje každá vrstva celkem tři podvrstvy. První je maskovaná multi-head attention vrstva, po které následuje další multi-head attention vrstva. Poslední podvrstvou je plně propojená dopředná vrstva. Stejně jako u kodéru i zde se využívá reziduálních propojeních a normalizací.

Maskování první podvrstvy umožňuje bezpečně provádět autoregresi, kdy se na všechny pozice $t+1$ až $t+n$ v čase t ve vstupní sekvenci nastaví $-\infty$, čímž se zabrání, aby informace z budoucích tokenů ovlivnily generování aktuálního tokenu.

Výstup kodéru je vstupem do druhé multi-head attention vrstvy.

Attention

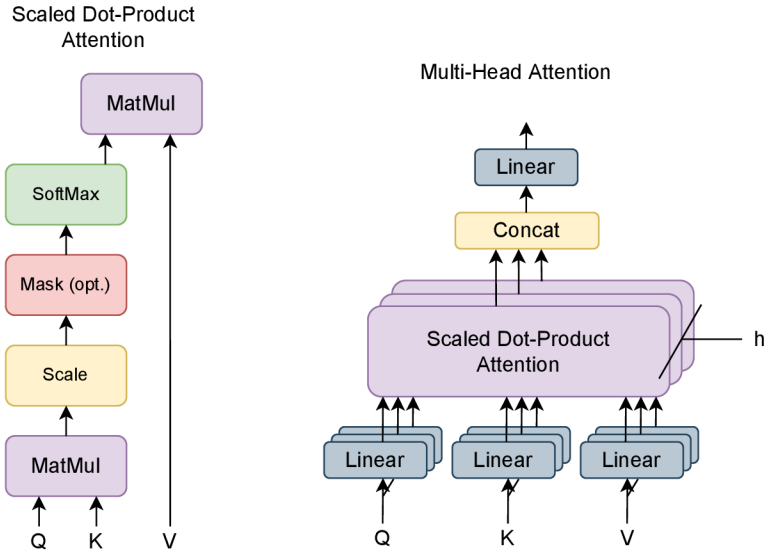
Attention je funkce, která na vstupu přijímá vektory *query* q, *key* k a *value* v. V případě, kdy q, k a v jsou totožné, označuje se tato funkce za *self-attention*. Vektory q a k mají stejný rozměr d_k. Vektor v má rozměr d_v. Při výpočtu attention se nejprve spočítá skalární součin mezi q a k, čímž se vypočítá *skóre*, které určuje, jak moc je k relevatní pro dané q. Skóre je pak škálováno pro udržení stabilních gradientů během trénování. V dalším kroku se na

skóre aplikuje funkce softmax k získání vah pro vektor v. Posledním krokem je provedení skalárního součinu mezi vypočítanými váhami a vektorem v.

V praxi se výpočet attention provádí na několika query současně, které jsou seskupeny do matice Q. Stejným způsobem jsou key a value seskupeny do matic K a V. Attention je pak dána následujícím vztahem:

$$Attention(Q, K, V) = softmax(\frac{QK^T}{\sqrt{d_k}})V \tag{2.1}$$

Tento typ attention funkce se nazývá *Scaled Dot-Product Attention*. Obrázek 2.2 znázorňuje proces výpočtu této attention funkce.



Obrázek 2.2: Attention (vlevo) a multi-head attention (vpravo), která se skládá z několika paralelních attention vrstev. Obrázek převzat z [29].

Multi-head Attention

Namísto provedení jedné attention funkce nad vstupními vektory, lze nejprve transformovat vektory q, k a v pomocí h lineárních vrstev, čímž vznikne h variant q, k a v. Ty jsou poté vstupem do h attention vrstev, které lze spočítat paralelně. Výstupy jsou pak konkatenovány do jednoho vektoru a transformovány další lineární vrstvou, aby byl zachován očekávaný rozměr výstupu (viz obrázek 2.2). Multi-head attention lze vyjádřit následujícím vztahem:

$$\begin{aligned} MultiHead(Q, K, V) &= Konkatenace(h_1, \ldots, h_h)W^O, \\ h_i &= Attention(QW_i^Q, KW_i^K, VW_i^V), \end{aligned} \tag{2.2}$$

kde lineární transformaci představují matice W_i^Q, W_i^K, W_i^V a W^O.

Poziční kódování

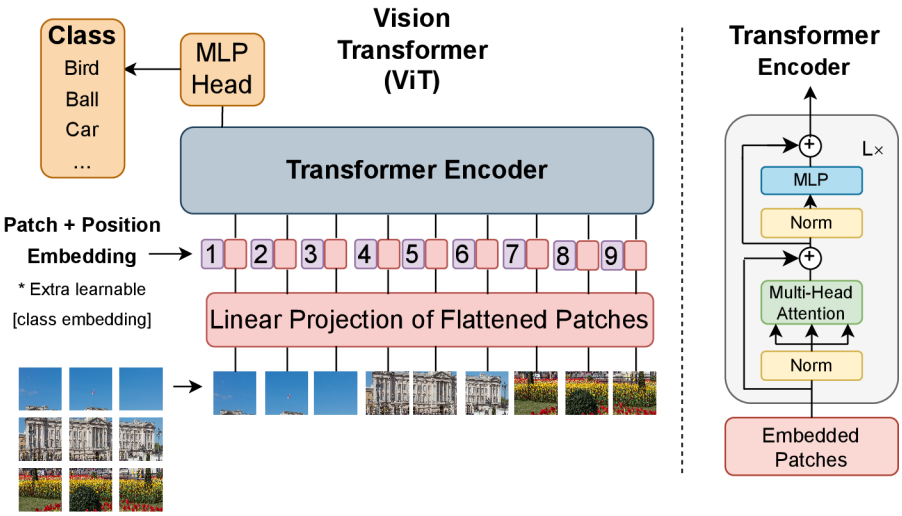
Před tím než vstupní matice embeddingů vstoupí do kodéru/dekodéru, je nejprve transformována tak, aby každý embedding nesl informaci o své pozici (viz obrázek 2.1). Jedna z možností, jak tuto informaci vložit do embeddingu, je použití následujících vztahů:

$$PK(pos, 2i) = sin(pos/10000^{2i/d_{model}}),$$
$$PK(pos, 2i+1) = cos(pos/10000^{2i/d_{model}}), \tag{2.3}$$

kde *pos* je pozice ze vstupní sekvence a i je indexem dimenze v embeddingu. Tyto vztahy vytvoří vektory o stejném rozměru jako měly vstupní embeddingy. Ty se pak přičtou ke vstupním embeddingům, čímž se zakomponuje informace o pozici do embeddingu. Kromě uvedených vztahů existuje řada dalších možností, jak vložit informaci o pozici [36].

2.2 Vision transformery

V reakci na úspěchy transformerů v oblasti zpracování přirozeného jazyka, se *Dosovitskiy et al.* rozhodli vyzkoušet v práci *An image is worth 16x16 words: Transformers for image recognition at scale* [4] architekturu transformerů na úlohách počítačového vidění. V této práci představili architekturu *Vision transformeru* (ViT) (viz obrázek 2.3).



Obrázek 2.3: Architektura modelu vision transformer. Obrázek převzat z [4].

ViT se drží co nejvíce architektury transformeru od *Vaswani et al.* [29]. Standardní transformer očekává na vstupu 1D sekvenci embeddingů. V případě 2D obrázků je vstupní obrázek $x \in \mathbb{R}^{H \times W \times C}$ transformován na sekvenci *narovnaných* (angl. flattened) 2D patchí $x_p \in \mathbb{R}^{N \times (P^2)C}$, kde (H, W) jsou výška a šířka vstupního obrázku, C je počet kanálů, (P, P) je šířka a výška každé patche a $N = HW/P^2$ je výsledný počet patchí. Tyto patche jsou pak namapovány do D dimenzí pomocí trénovatelné lineární projekce (vztah 2.4). Výsledné projekce se nazývají patch embeddingy.

$$\mathbf{z}_0 = [x_{class}; x_p^1\mathbf{E}; x_p^2\mathbf{E}, \dots x_p^N\mathbf{E}] + \mathbf{E_{pos}},$$
$$\mathbf{E} \in \mathbb{R}^{P^2C} \times D, \mathbf{E}_{pos} \in \mathbb{R}^{(N+1)\times D} \tag{2.4}$$

K sekvenci patch embeddingů autoři přidali speciální trénovatelný [*class*] token, ($\mathbf{z}_0^0 = \mathbf{x}_{class}$), jehož stav na výstupu kodéru $\mathbf{z}_L^0$ slouží jako reprezentace obrázku, podle vztahu 2.5:

$$\mathbf{y} = LayerNorm(\mathbf{z}_L^0), \tag{2.5}$$

kde $\mathbf{y}$ označuje konečnou reprezentaci vstupního obrázku.

Pro zachování informace o pozici jsou k patch embeddingům přidány poziční embeddingy. Výsledná sekvence embeddingů slouží jako vstup do kodéru, který je tvořen L vrstvami. Každá vrstva je tvořena multi-head self-attention funkcí, normalizacemi vrstev, reziduálními propojeními a *Multi-Layer perceptron* blokem, který jsou složen ze dvou vrstev s GELU aktivačními funkcemi.

Vstupní sekvence může být alternativně vytvořena i z map příznaků (angl. *feature maps*) z konvolučních sítí. Při tomto přístupu je projekce patch embeddingu $\mathbf{E}$ (viz vztah 2.4) aplikována na patche vyextrahované z map příznaků.

ViT se vyznačuje vysokou přesností a schopností generalizace při řešení úloh počítačového vidění. Zároveň přináší řadu výhod oproti tradičním konvolučním sítím, jako je například menší počet parametrů a lepší schopnost zachycování globálních vlastností obrazu.

ViT byl úspěšně použit v mnoha aplikacích počítačového vidění, jako jsou například klasifikace obrazů, segmentace obrazu a detekce objektů.

2.3 Triplet loss

Pro správné natrénování modelu pro re-identifikaci vozidel je potřeba vhodná loss funkce. Jednou z takových funkcí je triplet loss.

Triplet loss byla poprvé představena v práci [22], kde ji autoři použili pro natrénování mapování obrázků obličejů na vektory v euklidovském prostoru, ve kterém vzdálenosti těchto vektorů přímo odpovídaly míře podobnosti obličejů.

Při klasickém učení s učitelem je dopředu dán počet výstupů (tříd) modelu, který je většinou trénován pomocí cross-entropy loss funkce. Pro praktické určení podobnosti vozidel je ale potřeba dokázat pracovat i s proměnným počtem tříd.

Triplet loss umožňuje pracovat s novými třídami bez nutnosti přetrénování celého modelu. Dva objekty, které si jsou podobné, budou mít výsledné vektory, které vygeneruje model, vzdálenostně blízko u sebe, zatímco rozdílné objekty budou mít vektory od sebe vzdálené.

Triplet loss pracuje nad trojicí vstupů/obrázků, které se nazývají *anchor*, *positive* a *negative*. Anchor a positive náleží stejné třídě. Cílem je minimalizovat vzdálenost mezi výslednými vektory těchto dvou vstupů. Oproti tomu anchor a negative náleží rozdílným třídám a vzdálenost mezi jejich výstupními vektory je potřeba maximalizovat. Tedy pro trojici anchor, positive a negative musí platit následující vztah 2.6:

$$d(f_{anchor}, f_{positive}) + \alpha < d(f_{anchor}, f_{negative}) \tag{2.6}$$

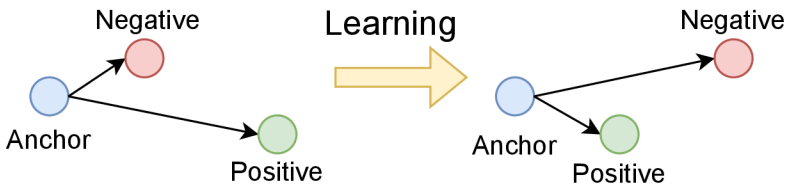
kde f_x je vektor reprezentující vstup x vygenerovaný trénovaným modelem, $d(x, y)$ je funkce pro výpočet vzdálenosti mezi vektory x a y, a α je *margin* hodnota mezi positive a negative páry.

Výsledná loss funkce, která se při trénování modelu minimalizuje, je pak dána následujícím vztahem 2.7:

$$\sum_{i}^{N} max(d(f_{anchor,i}, f_{positive,i}) - d(f_{anchor,i}, f_{negative,i}) + \alpha, 0) \tag{2.7}$$

kde N je kardinalita množiny všech možných trojic v trénovací sadě.

Pro re-identifikaci objektů je triplet loss velice často využívaná [7, 12, 18]. Obrázek 2.4 znázorňuje, jak triplet loss pracuje.



Obrázek 2.4: Triplet loss minimalizuje vzdálenost mezi anchor vstupem a positive vstupem, které jsou ze stejných tříd, a maximalizuje vzdálenost mezi anchor vstupem a negative vstupem, které jsou z různých tříd. Obrázek převzat z [22].

Výběr trojic

Na základě definice triplet loss lze rozdělit trojice vstupů do tří kategorií [32].

- **easy triplets** - musí platit podmínka 2.8, loss je rovna 0

$$d(anchor, positive) + \alpha < d(anchor, negative) \tag{2.8}$$

- **hard triplets** - musí platit podmínka 2.9, tedy negativní vstup je blíže anchor vstupu než pozitivní

$$d(anchor, negative) < d(anchor, positive) \tag{2.9}$$

- **semi-hard triplets** - musí platit podmínka 2.10, tedy negativní vstup není blíže k anchor vstupu než pozitivní, ale loss hodnota je stále pozitivní

$$d(anchor, positive) < d(anchor, negative) < d(anchor, positive) + \alpha \tag{2.10}$$

Tato definice kategorií závisí na tom, kde se nachází negativní vstup vzhledem k anchor a positive vstupu. Tyto kategorie lze označit jako *hard negatives*, *semi-hard negatives* a *easy negatives*.

Pro rychlou konvergenci loss funkce je zcela zásadní výběr trojic, které porušují podmínku 2.6. Existují dva možné přístupy, jak získat takové trojice.

Prvním přístupem je tzv. *Offline triplet mining*, kdy na začátku každé epochy se spočítají výstupní vektory pro všechny trénovací vstupy a pak jsou vybrány jen hard a semi-hard trojice. Tato technika ovšem není příliš efektivní, protože pro vygenerování trojic je nutné projít celým trénovacím datasetem a také je potřeba tyto vygenerované trojice opakovaně aktualizovat.

Druhým přístupem je tzv. *Online triplet mining*, který byl představen v práci [22]. Při tomto přístupu se trojice získávají za běhu pro každou batch vstupů. Při velikosti batch B lze spočítat B^3 trojic. Samozřejmě většina trojic nebude validních, protože například nebudou mít dva pozitivní a jeden negativní. Oproti offline miningu je tento přístup mnohem efektivnější, protože není potřeba procházet najednou celou trénovací sadu.

Kapitola 3

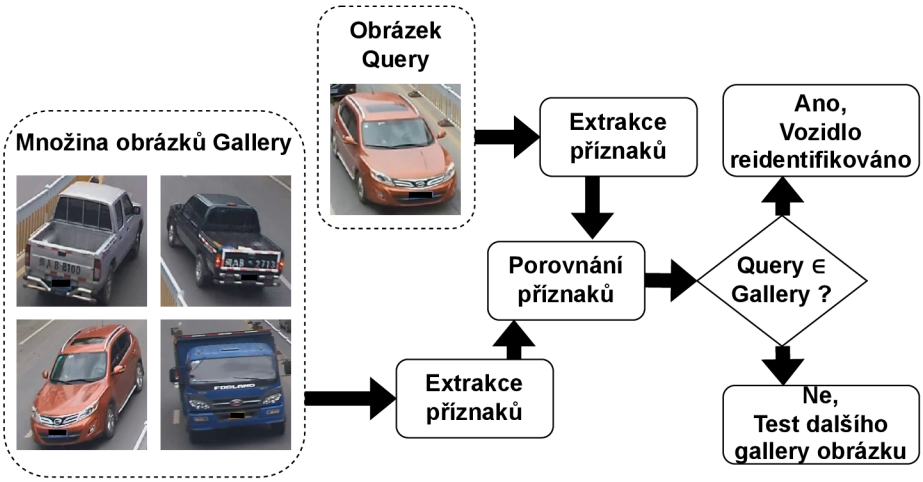
Úloha re-identifikace

Re-identifikace vozidel (ReID) je úloha počítačového vidění, která se zaměřuje na nalezení stejné identity vozidla na různých snímcích z různých kamer. V poslední době se této úloze věnuje stále větší pozornost vzhledem k jejímu širokému uplatnění v mnoha oblastech, jako je analýza městské dopravy, řízení dopravy atd. V rámci re-identifikace vozidel existují dva hlavní přístupy: *image-to-image* re-identifikace a *image-to-tracklet* re-identifikace [17].

Formálně lze proces image-to-image re-identifikace definovat jako úkol nalezení shody mezi *query* obrázkem a obrázků z množiny *gallery*, která obsahuje několik různých identit, na základě vizuálních podobností. Natrénovaný model zpracovává obrázky vozidel a extrahuje jejich vlastnosti, které se pak porovnávají pomocí metrik pro podobnost či vzdálenost [34]. Tedy, pro query obrázek Q se hledá takový obrázek z množiny gallery, aby platilo následující:

$$\begin{aligned} &T = \arg_{T_i} \min Dist(f(T_i), f(Q)), \\ &T_i \in \mathcal{T}, \end{aligned} \tag{3.1}$$

kde $\mathcal{T} = \{T1, \ldots, T_N\}$ je gallery množina o N obrázcích a funkce f je transformační funkce, která ze vstupního obrázku vytvoří vektor popisující daný obrázek. Funkce $Dist$ je distanční metrika [34]. Obrázek 3.1 znázorňuje celý proces re-identifikace.



Obrázek 3.1: Proces re-identifikace. Obrázek adaptován dle originálu [34].

V přístupu image-to-tracklet se místo porovnání jednotlivých obrázků porovnávají celé *tracklety* vozidel. Tracklet je sekvence snímků vozidla zachycených jednou nebo více kamerami v průběhu času. Reprezentace identity vozidla je pak dána jako průměr příznaků jednotlivých snímků v trackletu [17].

Re-identifikace vozidel je náročná z mnoha důvodů. Jedním z nejzásadnějších problémů je fakt, že vozidlo zachycené z různých úhlů pohledu má obvykle výrazně odlišný vzhled [13]. Jak je vidět na obrázku 3.2, snímky vozidel získané z různých úhlů pohledu mají zjevně odlišný vzhled. Na druhou stranu vozidla s různými identitami budou mít v některých případech velmi podobný vzhled, jak ukazuje obrázek 3.3. ReID vozidel je tedy náročná klasifikační úloha s vysokou vnitro-třídní variabilitou a vysokou podobností mezi třídami. Proto bude mít pro re-identifikaci vozidel zásadní význam učení rozlišovacích příznaků, které dokážou rozlišit různé identity, ale jsou neměnné v závislosti na úhlu pohledu [13].

Obrázek 3.2: Vzhled stejných vozidel z různých úhlů. Vzhled se viditelně mění. Barva vozidla se může lehce měnit s různým osvětlením. Obrázek převzat z [13].

Obrázek 3.3: Vzhledy dvou různých vozidel si mohou být velice podobné, jsou-li zachyceny ze stejného úhlu pohledu. Obrázek převzat z [13].

Zásadním úkolem re-identifikace je tedy nalezení takové funkce f, která dokáže dostatečně dobře vytvořit vhodnou reprezentaci vstupních obrázků. K tomu lze využít strojové učení a modely na něm založené.

Často používaným postupem pro re-identifikaci objektů je nadefinování vhodné loss funkce pro natrénování konvoluční sítě (např. ResNet [6]) jako *backbone* modelu [7]. Tato backbone pak slouží k extrakci příznaků z obrázků. Nejpoužívanější loss funkce pro re-identifikaci jsou cross-entropy loss (někdy též ID loss) [40] a triplet loss [14]. Pro lepší kombinaci ID loss a triplet loss navrhl *Luo et al.* [18] *BNNeck*, která normalizuje výstupní

příznaky před vstupem do plně propojené vrstvy. *Sun et al.* navrhl *Circle loss*, která je méně citlivá na výběr trojic anchor, positive a negative (viz podkapitolu 2.3).

Další metody extrahují příznaky z různých významných oblastí v obrázku. Významné oblasti jsou části obrázků, ve kterých se nacházejí klíčové informace pro re-identifikaci vozidel, například číslo registrační značky, tvar nebo nálepky na předním okně vozidla. Obrázek 3.4 ukazuje, jak tyto oblasti mohou vypadat. Správný výběr významných oblastí má zásadní vliv na úspěšnost těchto metod, a je proto žádoucí, aby byly detekovány ty oblasti, které nesou nejinformativnější hodnotu. Neinformativní oblasti mohou vést ke špatným a nestabilním výsledkům [13]. Metody PCB [25], MGN [30], AlignedReID++ [19] a SAN [21] rozdělují vstupní obrázek do několika *pruhů* a z každého tohoto pruhu extrahují lokální příznaky.

Obrázek 3.4: Vozidla, která mají globální vzhled stejný, ale liší se v lokálních oblastech, které rozlišují identitu vozidla.

Vytvoření anotací pro významné oblasti bývá velmi náročné [13]. Pro překonání nedostatků anotací lze použít tzv. *Weakly supervised* metody, a to jen s mírnou ztrátou přesnosti. Například v práci od *Khorramshahi et al.* [11] vytvořili model SAVER, který automaticky zvýrazňuje významné oblasti na obrázku vozidla pomocí *Variational Auto-Encoderu* (VAE). VAE je schopno vygenerovat rekonstrukci vstupního obrázku bez klíčových detailů. Tato rekonstrukce se odečte od vstupního obrázku, čímž se zvýrazní významné oblasti na vozidle.

Aby překonaly změny v osvětlení, pozice, orientaci atd. v obrázcích zachycenými vícero kamerami, některé práce zakomponovaly dodatečné informace, jako je ID kamery nebo úhel pohledu, z jakého byl objekt zachycen. Tyto příznaky jsou neměnné. Například *Camera-based Batch Normalization* (CBN) [42] se zaměřuje na normalizaci dat, která pocházejí z různých kamer, aby byla odstraněna rozdílnost v rozložení dat mezi různými kamerami. Tento proces umožňuje modelům učit se vytvářet příznaky, které jsou invariantní vůči kamerám. Podobně v práci *A Dual-Path Model With Adaptive Attention For Vehicle Re-Identification* [10] od *Khorramshahi et al.* navrhli model, který ke globálním příznakům přidává lokální příznaky, které získává z odhadnutých klíčových bodů na vozidle. Z těchto bodů dále odhadují úhel pohledu, ze kterého bylo vozidlo nasnímáno. Všechny tyto příznaky

jsou poté spojeny, aby vytvořily vektor popisující vstupní obrázek vozidla. VANet z práce *Vehicle Re-identification with Viewpoint-aware Metric Learning* [3] využívá dvě větve pro extrakci odlišných příznaků, které poté využívá pro naučení se tzv. *viewpoint-aware* metrik. Model nejprve odhadne, z jakých úhlů pohledu byly obrázky s vozidly zachyceny. Jestliže dvojice obrázků byla zachycena v podobném úhlu, použije se funkce f_s pro převod obou vstupních obrázků do prostoru příznaků a poté se spočítá vzdálenost mezi nimi a určí se, zda se jednalo o stejnou identitu vozidla. Pokud vozidla nebyla zachycena ze stejných úhlů, použije se funkce f_d.

3.1 Řešení pomocí vision transformerů

Vision Transformer (ViT) byl poprvé využit v práci od *Dosovitskiy et al.* [4] pro úlohu klasifikace (viz podkapitolu 2.2). Tento model dosáhl uspokojivých výsledků na rozsáhlých a komplexních datasetech tím, že rozdělil vstupní obraz na menší patche a zachytil tak globální informace skrze sekvenci těchto patchí [4].

Trénování ViT vyžaduje rozsáhlý dataset. K překonání tohoto problému navrhli *Touvron et al.* model *Data-efficient image Transformer* (DeiT) [28], který zavedl strategii učitel-student specifickou pro transformery ke zrychlení procesu trénování ViT bez nutnosti rozsáhlého datasetu.

Transformery byly úspěšně použity nejen pro klasifikaci, ale také pro řešení dalších úloh počítačového vidění, jako jsou detekce objektů [1], sémantická segmentace [39] a zpracování obrazu [2]. Model *Image Processing Transformer* [2], založený čistě na transformerech, který byl předtrénován na datasetu ImageNet, dosáhl state-of-the-art výsledků na několika úlohách zpracování obrazu, jako je zvýšení rozlišení a odstranění šumu a deště z obrazu.

Úspěšná se ukázala i kombinace konvolučních neuronových sítí spolu s vision transformery. V práci od *Guo et al.* [5] autoři vytvořili pomocí škálování celou rodinu modelů nazvaných *CMT*, které využívají konvoluce spolu s attention mechanismem. Tyto modely dosáhly výborných výsledků na úlohách klasifikace obrázků, detekce objektů i segmentace instancí.

Jeden z prvních pokusů o použití vision transformerů k re-identifikaci vozidel podnikl *He et al.* v práci *TransReID: Transformer-based Object Re-Identification* [7]. Zde autoři vytvořili model založený čistě na vision transformeru. K tomuto modelu přidali možnost zakomponovat informace o ID kamery či úhlu pohledu, ze kterého byl vstupní obrázek zachycen. Dalším pokusem o využití transformeru k re-identifikaci vozidel byla práce od *Lian et al.* [13]. Autoři navrhli model o třech větvích, kde jedna z nich využívala attention mechanismu, druhá byla založena na konvolučních sítích a pomocí třetí extrahovali další pomocné informace (barva, typ vozidla atd.).

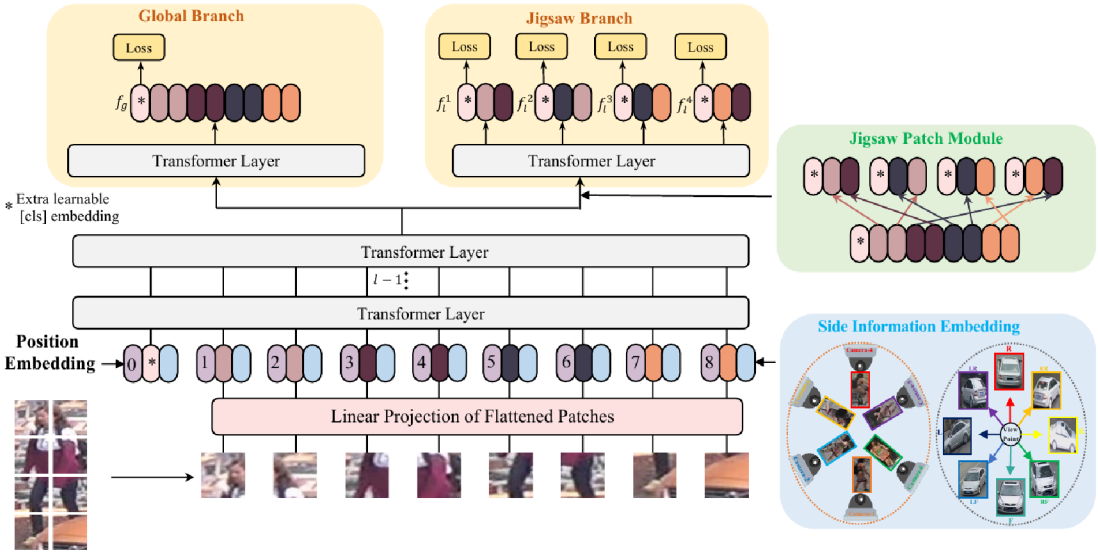
Je tedy patrné, že vision transformery mají velký potenciál dosáhnout state-of-the-art výsledků v mnoha oblastech počítačového vidění. V následujících podkapitolách detailně popíši vybrané modely, se kterými jsem experimentoval.

3.2 TransReID

V práci *TransReID: Transformer-based Object Re-Identification* [7] od *He et al.*, autoři navrhli robustní model založený čistě na vision transformeru, pojmenovaný *TransReID*.

Celý model vychází z modelu ViT (viz podkapitolu 2.2) pro klasifikaci obrazů obohacený o tzv. *jigsaw patch modul* (JPM) a *side information embeddings* (SIE). JPM má za cíl

přeuspořádat patch embeddingy pomocí operací *shift* a *shuffle* a tím vygenerovat robustní příznaky s lepšími rozlišovacími schopnostmi. SIE připojuje ke vstupním embeddingům informaci o ID kamery a z jakého pohledu bylo vozidlo zachyceno. Celý model TransReID je na obrázku 3.5.



Obrázek 3.5: Model TransReID. SIE (bledě modré) zakomponovává nevizuální informace (ID kamery, pohled na vozidlo) do embeddingů, které se připojují k patch a pozičním embeddingům. Poslední vrstva je tvořena dvěma nezávislými transformer vrstvami. Jedna je standardní transformer vrstva jako z modelu ViT (označeno jako global branch) a druhá obsahuje JPM, který přeskládává patche a seskupuje je do skupin. Tyto skupiny jsou pak vstupem do tzv. *jigsaw* větve, kde se učí lokální příznaky. Obrázek převzat z [7].

Vstupní obraz $x \in \mathbb{R}^{H \times W \times C}$, kde H, W, C jsou výška, šířka a počet kanálů, je rozdělen na N patchí o pevně dané velikosti $\{x_p^i | i = 1, 2, \ldots N\}$. Ke vstupní sekvenci je ještě připojen trénovatelný embedding token [cls] označený jako x_{cls}. Tento token slouží jako reprezentace globálního příznaku f. Prostorová informace je zahrnuta přidáním trénovatelných pozičních embeddingů. Vstupní sekvence lze pak vyjádřit jako:

$$\mathcal{Z}_0 = [x_{cls}; \mathcal{F}(x_p^1); \mathcal{F}(x_p^2); \ldots; \mathcal{F}(x_p^N)] + \mathcal{P}, \tag{3.2}$$

kde $\mathcal{Z}_0$ je vstupní sekvence embeddingů, $\mathcal{P} \in \mathbb{R}^{N+1} \times D$ jsou poziční embeddingy a $\mathcal{F}$ je lineární projekce mapující patche vstupního obrazu do D dimenzí. Pro natrénování reprezentace příznaků je vytvořeno l transformer vrstev.

Model je optimalizován pomocí ID loss a triplet loss (viz podkapitolu 2.3).

Překrytí patchí

Aby zachovali informace o strukturách okolo vytvořených patchí obrazu, rozhodli se autoři pro použití tzv. *sliding window* k vygenerování patchí s překrývajícími se pixely. Při velikosti kroku S a velikosti patche P, je velikost překrývající oblasti $(P - S) \times P$. Vstupní obrázek s rozměry $H \times W$, je poté rozdělen na N patchí, podle vztahu 3.3.

$$N = N_H \times N_W = \lfloor \frac{H + S - P}{S} \rfloor \times \lfloor \frac{W + S - P}{S} \rfloor, \tag{3.3}$$

kde $\lfloor \cdot \rfloor$ je funkce zaokrouhlení směrem dolů a platí, že $S < P$. Čím menší S, tím více patchí vznikne.

Jigsaw Patch Module

Cílem JPM je naučení se lokálních příznaků. JPM promíchá patch embeddingy a pak je rozdělí do několika skupin, z nichž každá obsahuje několik náhodně vybraných patch embeddingů. Embeddingy jsou promíchány pomocí operací shift a shuffle. Za předpokladu, že příznaky vstupující do poslední vrstvy jsou označeny jako $\mathcal{Z}_{l-1} = [z^0_{l-1}; z^1_{l-1}, z^2_{l-1}, \ldots, z^N_{l-1}]$, je pak tato sekvence promíchána následovně:

- Shift: Prvních m patchí, [cls] token vyjímaje, je posunuto na konec. Ze sekvence $[z^1_{l-1}, z^2_{l-1}, \ldots, z^N_{l-1}]$ se stane v m krocích $[z^{m+1}_{l-1}, z^{m+2}_{l-1}, \ldots, z^N_{l-1}, z^1_{l-1}, z^2_{l-1}, \ldots, z^m_{l-1}]$.
- Shuffle: Posunuté patche jsou promíchány do k skupin. Výsledkem je pak přeuspořádaná sekvence embeddingů $[z^{x_1}_{l-1}, z^{x_2}_{l-1}, \ldots, z^{x_N}_{l-1}], x_i \in [1, N]$.

Celková loss funkce je pak spočítána následovně:

$$\mathcal{L} = \mathcal{L}_{ID}(f_g) + \mathcal{L}_T(f_g) + \frac{1}{k}\sum_{j=1}^{k}(\mathcal{L}_{ID}(f^j_l) + \mathcal{L}_T(f^j_l)), \tag{3.4}$$

kde f^j_l jsou lokální příznaky získané pomocí JPM, f_g jsou globální příznaky z globální větve modelu. $\mathcal{L}_{ID}$ a $\mathcal{L}_T$ jsou ID loss, respektive triplet loss (viz vztah 2.7).

Během inference se globální a lokální příznaky konkatenují $[f_g, f^1_l, f^2_l, \ldots, f^k_l]$, aby vytvořili finální reprezentaci.

Side Information Embeddings

Pro překonání variability mezi scénami z různých kamer a různých úhlů pohledu na vozidlo, autoři přidali k modelu mechanismus SIE, který přidává nevizuální informace ke vstupním embeddingům.

SIE pro ID kamer se inicializuje jako $S_C \in \mathbb{R}^{N_C \times D}$, kde N_C je celkový počet kamer. Je-li ID kamery označeno jako r, pak jeho embedding je značen jako $S_C[r]$. Na rozdíl od patch embeddingů, které se mezi patchemi mění, embeddingy $S_C[r]$ zůstávají pro všechny patche obrázku stejné. Dále, je-li k dispozici informace o úhlu pohledu q na vozidlo, ať už algoritmem pro jeho odhad, nebo anotací, lze tuto informaci zakódovat jako $S_V[q]$ pro všechny patche obrázku, kde $S_V \in \mathbb{R}^{N_V \times D}$ a N_V je počet ID úhlů pohledu (viewpoint).

Společně se pak S_C a S_V kódují jako $S_{(C,V)} \in \mathbb{R}^{(N_C \times N_V) \times D}$. Vstupní sekvence embeddingů spolu s ID kamery r a ID úhlu pohledu q má pak následovný tvar:

$$\mathcal{Z}^{'}_0 = \mathcal{Z}_0 + \lambda S_{(C,V)[r*N_V+q]}, \tag{3.5}$$

kde $\mathcal{Z}_0$ je vstupní sekvence definovaná vztahem 3.2 a λ je hyperparametr k balancování váhy SIE.

Autoři tento model testovali nejen na datasetech pro re-identifikaci vozidel, ale i pro re-identifikaci osob. Pro ReID vozidel využili datasety VeRi-776 (viz podkapitolu 4.1) a VehicleID (viz podkapitolu 4.2).

V rámci soutěže *AI City Challenge 2021* (AICITY21) [20], kterou pořádala společnost *NVIDIA* v roce 2021, autoři otestovali TransReID na datasetu CityFlow. Autoři ovšem

neuvedli výsledek na testovacích datech pro samotné TransReID. Ve své práci zkombinovali výsledky z celkem 16 modelů, včetně TransReID, a až konečný výsledek zveřejnili [17].

Abych získal porovnání s jejich natrénovaným TransReID modelem na datasetu CityFlow, obstaral jsem si autory natrénované váhy[1] modelu a vyhodnotil jsem jej sám za použití evaluačního systému od pořadatelů soutěže.

Tabulka 3.1 shrnuje výsledky autorů s různými konfiguracemi modelu a mnou vyhodnocený model TransReID na datasetu CityFlow, za použití vah natrénovaných autory.

Model	VeRi-776		VehicleID		CityFlow	
	mAP	Rank1	Rank1	R5	mAP	Rank1
Baseline	0,782	96,5	82,3	96,1	**0,565**	**69,7**
TransReIDv	0,796	97,0	83,6	97,1	—	—
TransReIDv,c	0,806	96,9	—	—	—	—
TransReID$*,v$	0,805	96,8	**85,2**	**97,5**	—	—
TransReID$*,v,c$	**0,820**	**97,1**	—	—	—	—

Tabulka 3.1: Dosažené výsledky autorů modelu TransReID na datasetech pro re-identifikaci vozidel. Symbol * znamená, že patche ze vstupního obrázku se tvoří pomocí sliding window. Symboly v a c znamenají, že modely využívají viewpoint značky, respektive ID kamer v modulu SIE.

3.3 Convolutional Neural Networks Meet Vision Transformers

Guo et al. se ve své práci *Convolutional Neural Networks Meet Vision Transformers* [5] pokusili zvýšit úspěšnost vision transformerů jejich spojením s konvolučními neuronovými sítěmi. Hlavní myšlenka tohoto spojení byla využít schopnost vision transformerů zachytit závislosti na větší vzdálenost spolu se schopností konvolučních sítí extrahovat lokální informace. Jejich dalším cílem bylo snížit rozdíl ve výpočetní náročnosti mezi vision transformery a konvolučními sítěmi.

Autoři navrhli hybridní síť, jejímž škálováním vytvořili celou rodinu modelů zvanou CMT. Na obrázku 3.6 je příklad jednoho takového modelu.

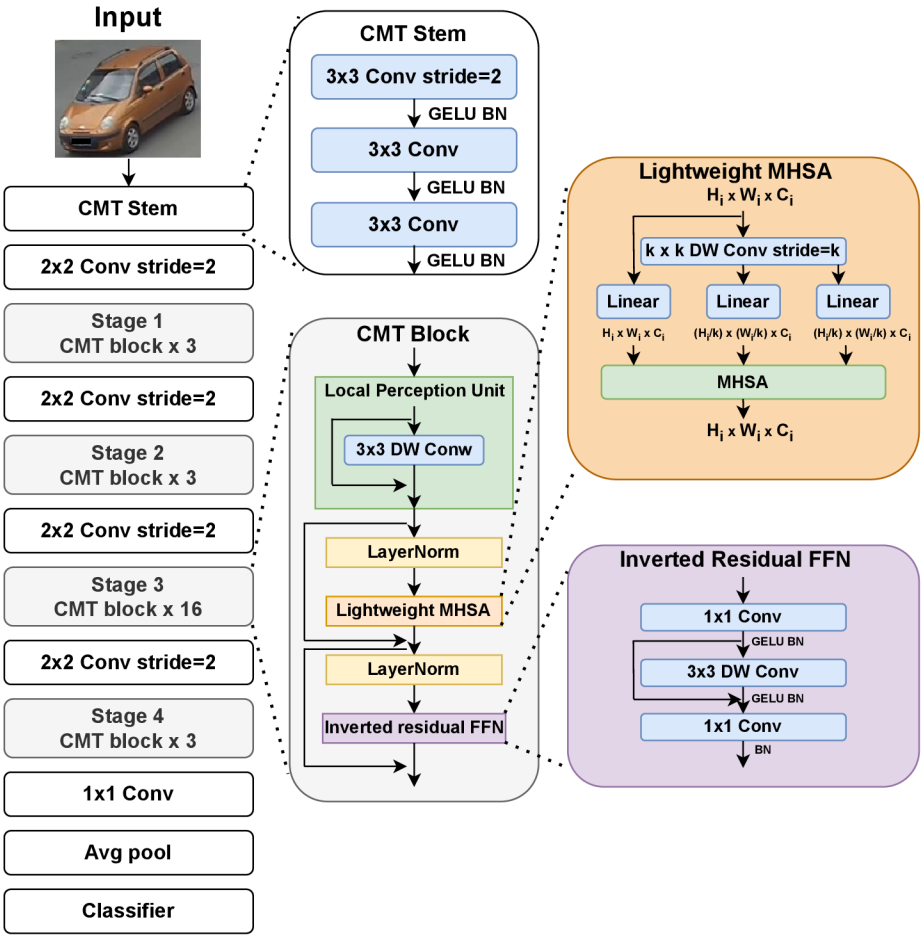
Jak je vidět na obrázku 3.6, vstupní obrázek nejprve prochází vrstvou *Convolution stem*, kde dochází k extrakci *fine-grained* příznaků. Ty dále procházejí čtyřmi fázemi modelu, kde se učí reprezentace vstupního obrázku.

Každé fázi předchází *Patch embedding layer*, která je složena z konvoluce a normalizace samotné vrstvy, a má za úkol snížit počet vstupních příznaků (dvojnásobné podvzorkování) a pomocí projekce zdvojnásobit jejich dimenzi.

Dále každá fáze je složena z několika CMT bloků. CMT blok je vylepšená varianta klasického transformer bloku o *depth-wise* konvoluci pro lepší extrakci lokálních příznaků. Oproti modelu ViT [4] si příznaky vygenerované v první fázi CMT (stage 1) dokáží zachovat větší rozlišení ($H/4 \times W/4$ oproti $H/16 \times W/16$ ve ViT). Každý CMT blok generuje mapu příznaků o různých velikostech.

CMT blok je složen z *Local perception unit* (LPU), *lightweight multi-head self-attention* (LMHSA) a *inverted residual feed-forward network*. LPU se snaží vyřešit limitace pozičního

[1] Váhy modelu dostupné na https://github.com/michuanhaohao/AICITY2021_Track2_DMT



Obrázek 3.6: Architektura modelu CMT-S. Obrázek převzat z [5].

kódování z vision transformerů. Autoři argumentují, že poziční kódovaní přiřazené každé patchi vstupního obrázku, které mělo za cíl využít pořadí vstupních tokenů, nepomáhá v rámci invariace vůči translaci, která je samozřejmě očekávána v úlohách počítačového vidění. Operace LPU je definována následovně:

$$LPU(X) = DWConv(X) + X, \tag{3.6}$$

kde $X \in \mathbb{R}^{H \times W \times d}$, $H \times W$ je rozlišení vstupu z aktuální fáze, d je rozměr příznaků a $DWConv(\cdot)$ je depth-wise konvoluce.

Lightweight Multi-head Self-attention je modul, který má za cíl snížit výpočetní náročnost klasického attention modulu (viz podkapitolu 2.1). Před samotnou operací attention je aplikována $k \times k$ depth-wise konvoluce s velikostí kroku k pro snížení velikosti K a V. Tedy $K' = DWConv(K)$ a $V' = DWConv(V)$. Dále je ke každému self-attention modulu přidán poziční bias B. Výsledná lightweight attention je definována jako:

$$LightAttn(Q, K, V) = Softmax(\frac{QK'}{\sqrt{d_k}} + B)V', \tag{3.7}$$

kde $B \in \mathbb{R}^{n \times \frac{n}{k^2}}$ je trénovatelné a inicializované náhodně. LMHSA modul je definován pro h *hlav*, tedy h Lightweight attention funkcí je aplikováno na vstup. Každá hlava vyprodukuje

sekvenci o velikosti $n \times \frac{d}{h}$. Tyto sekvence jsou pak konkatenovány do sekvence o velikosti $n \times d$.

Inverted Residual Feed-forward Network je definována následovně:

$$\begin{aligned} IRFFN(X) &= Conv(\mathcal{F}(Conv(X))), \\ \mathcal{F} &= DWConv(X) + X \end{aligned} \tag{3.8}$$

Depth-wise konvoluce je použita pro extrakci lokálních příznaků se zanedbatelnou výpočetní náročností. Motivace k přidání reziduálního propojení je stejná jako u reziduálních sítí. Tedy pomáhá proti mizejícímu či explodujícímu gradientu ve vrstvách.

Formálně tedy lze CMT blok definovat následovně:

$$\begin{aligned} Y_i &= LPU(X_{i-1}), \\ Z_i &= LMHSA(LN(Y_i)) + Y_i, \\ X_i &= IRFFN(LN(Z_i)) + Z_i, \end{aligned} \tag{3.9}$$

kde Y_i a Z_i jsou výstupní příznaky z LPU, respektive LMHSA modulu pro blok i. LN je normalizace vrstvy.

Pomocí škálování vznikly celkem čtyři modely. Jsou to CMT-ti, CMT-xs, CMT-s a CMT-b. Liší se počtem CMT bloků v každé fázi a velikostí vstupního obrázku. CMT-ti přijímá na vstupu obrázek o rozměru 160×160, CMT-xs přijímá obrázek o rozměru 192×192, CMT-s přijímá obrázek o rozměru 224×224 a CMT-b přijímá obrázek o rozměru 256×256.

Autoři provedli sadu experimentů nad několika úlohami z počítačového vidění, jako je klasifikace, detekce objektů či segmentace instancí. Pro klasifikaci natrénovali a vyhodnotili model na datasetu ImageNet. Na detekci objektů a segmentaci využili dataset COCO. Tabulka 3.2 shrnuje jejich dosažené výsledky na datasetu ImageNet v porovnání s dalšími state-of-the-art modely. Tabulky 3.3 a 3.4 shrnují výsledky na datasetu COCO při úloze detekce objektů, respektive segmentace.

Model	Top-1 Acc.	Top-5 Acc.
EfficientNet-B1	74,1%	94,4%
EfficientNet-B4	82,9%	96,4%
EfficientNet-B7	84,3%	97,0%
ResNet-50	76,2%	92,2%
DeiT-S	79,8%	—
T2T-ViT-19	81,2%	—
CMT-ti	73,1%	94,5%
CMT-xs	81,8%	95,8%
CMT-s	83,5%	96,6%
CMT-b	84,5%	96,9%
CMT-l	**84,8%**	**97,1%**

Tabulka 3.2: Výsledky CMT na datasetu ImageNet při úloze klasifikace.

Z tabulek je zjevné, že CMT dosahuje state-of-the-art výsledků na základních úlohách počítačového vidění. Re-identifikaci vozidel autoři nezačlenili do svých experimentů, a proto nebudu schopen nabídnout porovnání svých výsledků s jejich výsledky.

Model	mAP	AP_{50}	AP_{75}
ResNet-101	38,5	57,6	41,0
RelationNet++	39,4	58,2	42,5
PVT-S	40,4	61,3	43,0
CMT-s	**44,3**	**65,5**	**47,5**

Tabulka 3.3: Výsledky CMT na datasetu COCO při úloze detekce objektů.

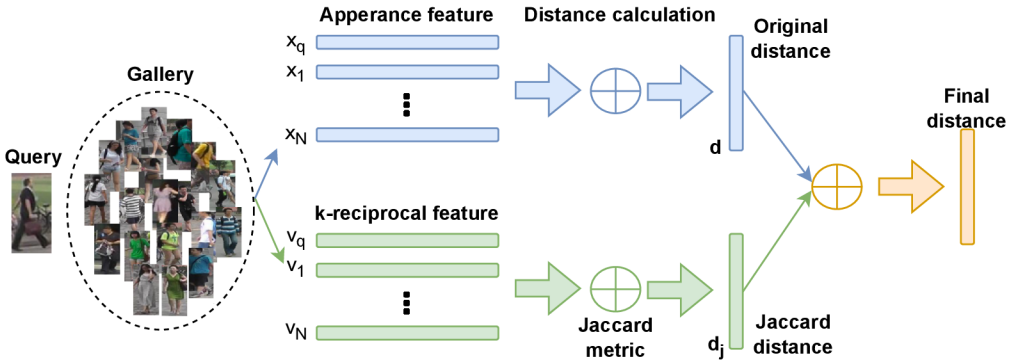
Model	AP_{box}	$AP_{box,\ 50}$	$AP_{box,\ 75}$
ResNet-101	40,0	60,5	44,0
Twins-SVT-S	42,7	65,6	46,7
PVT-S	40,4	62,9	46,7
CMT-s	**44,6**	**66,8**	**48,9**

Tabulka 3.4: Výsledky CMT na datasetu COCO při úloze segmentace instancí.

3.4 Reranking

Reranking je technika, která se často používá k vylepšení výsledků re-identifikace vozidel. Základní myšlenkou je přeuspořádání pořadí výsledků získaných z původního hodnocení podobnosti tak, aby byly výsledky přesnější a relevantnější. Reranking se často provádí pomocí dodatečných informací nebo vztahů mezi obrázky v datasetu [41].

Obecně lze reranking rozdělit do několika kroků. V prvním kroku se získá počáteční uspořádání gallery obrázků vzhledem k query obrázku. Dále se pro gallery i query obrázky získají dodatečné informace či vztahy mezi obrázky. Jedním z možných přístupů je k-vzájemných nejbližších sousedů [41]. Na základě těchto dodatečných informací se upraví počáteční uspořádání, které by mělo být přesnější. Obrázek 3.7 znázorňuje postup rerankingu.



Obrázek 3.7: Princip rerankingu s k-vzájemnými nejbližšími sousedy. Obrázek převzat z [41].

Kapitola 4

Datové sady

V této kapitole popisuji dostupné datové sady, které lze využít k re-identifikaci vozidel. Minimální datová sada musí obsahovat snímky vozidel rozdělené do dvou disjunktních množin. Jedna z těchto množin slouží k natrénování modelu a druhá k jeho testování. Testovací sada je dále rozdělena na dvě další části. První částí jsou *query* obrázky, na kterých jsou vozidla, která je třeba identifikovat. Druhou částí jsou *gallery* obrázky, které se porovnávají s daty z query množiny.

Každý obrázek vozidla musí mít přiřazenou identitu. Další informace, které datasety mohou obsahovat, jsou atributy vozidel, např. barva nebo typ, ID kamery nebo úhel, ze kterého bylo vozidlo nasnímáno.

4.1 VeRi-776

VeRi-776 je volně dostupný dataset vytvořený pro re-identifikaci vozidel [16]. Dataset je tvořen snímky vozidel z reálných situací zachycených monitorovacími kamerami. Sestaven je z 51 035 snímků se 776 unikátními vozidly. Každé vozidlo bylo zachyceno 2 až 18 různými kamerami, s různým rozlišením, osvícením a zastíněním.

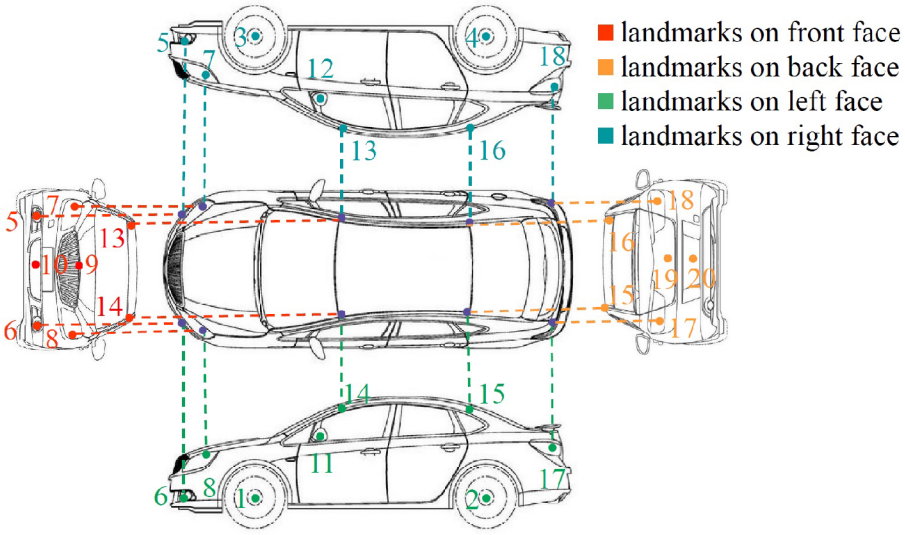
Dataset je rozdělen na trénovací, gallery a query data. Trénovací data jsou složena z 37 778 snímků, gallery z 11 579 a query z 1 678. Každému trénovacímu snímku jsou přiřazeny dodatečné informace, a sice ID vozidla, ID kamery, která snímek pořídila, barva vozidla a typ vozidla.

Ukázková data z datasetu VeRi-776 jsou na obrázku 4.1.

Obrázek 4.1: Ukázka datasetu VeRi-776.

V práci *Orientation Invariant Feature Embedding and Spatial Temporal Regularization for Vehicle Re-identification* od *Wang et al.* [31] autoři manuálně vytvořili pro dataset VeRi-

776 seznam klíčových bodů[1], které považovali za nejvýznamnější. Celkem specifikovali 20 klíčových bodů na celém vozidle (viz obrázek 4.2).



Obrázek 4.2: Klíčové body na vozidle z datasetu VeRi-776. Obrázek převzat z [31].

4.2 PKU VehicleID

Dataset PKU VehicleID je sestaven z celkem 221 567 snímků zachycujících 26 328 unikátních vozidel [15]. Ke každému snímku je přiřazeno ID vozidla a barva vozidla. Dataset obsahuje také 10 319 manuálně anotovaných vozidel (celkem 90 196 snímků) s jejich příslušným typem vozidla (např. MINI-cooper, Audi A6L atd.).

Trénovací data jsou tvořena 13 164 vozidly (113 346 snímků). Pro testovací účely má VehicleID tři různé sady. *Small* o 800 identitách (800 snímků), *medium* o 1600 identitách (1600 snímků) a *large* o 2400 identitách (snímků). Pro každý query snímek je v gallery sadě pouze jedna shodná identita.

Příklad snímků z datasetu VehicleID je na obrázku 4.3.

Obrázek 4.3: Ukázka datasetu VehicleID.

[1]Anotace klíčových bodů pro dataset VeRi-776 dostupné na https://github.com/Zhongdao/VehicleReIDKeyPointData

4.3 CityFlowV2-ReID

V soutěži *AI City Challenge*[2] z roku 2021 bylo představeno celkem 5 výzev, ve kterých se, za pomocí AI, řešily úlohy z počítačového vidění spojené s dopravou ve městech. Jednou z těchto výzev byla re-identifikace vozidel, ke které byl zveřejněn *CityFlowV2-ReID* dataset [20, 27].

Tento dataset je složen ze dvou částí. První část je tvořena daty z reálných dopravních situací ve městech (celkem 85 058 snímků) s 880 unikátními vozidly a druhá část je tvořena syntetickými daty (celkem 192 150 obrázků). Trénovací sada z první části má 31 238 snímků, gallery sada 52 717 a query seda 1 103. Každému snímku je přiřazena informace o identitě vozidla na snímku a identitě kamery, která snímek pořídila. Syntetická část dat je tvořena datasetem *VehicleX*[3] [33, 26], který byl vytvořen pomocí stejně pojmenovaného enginu *VehicleX*. Tento engine byl vytvořen za použití 3D enginu *Unity*[4]. Syntetická data jsou určena jen pro trénování. Celkem je na nich označeno 1 362 identit vozidel. Ke každému obrázku je přiřazeno ID vozidla, ID kamery, barva a typ vozidla. Dále jsou přítomny parametry osvětlení a kamery z Unity scény.

Ukázková data z datasetu CityFlow jsou na obrázku 4.4 a z datasetu VehicleX na obrázku 4.5.

Obrázek 4.4: Ukázka datasetu CityFlow.

Obrázek 4.5: Ukázka datasetu VehicleX.

Testovací část datasetu neobsahuje ground-truth identity vozidel. Není tedy možné vyhodnotit natrénované modely lokálně. Pro vyhodnocení modelů byl vytvořen evaluační systém[5], ke kterému je nutné mít vytvořený a schválený účet.

[2] https://www.aicitychallenge.org/2021-ai-city/

[3] https://github.com/yorkeyao/VehicleX

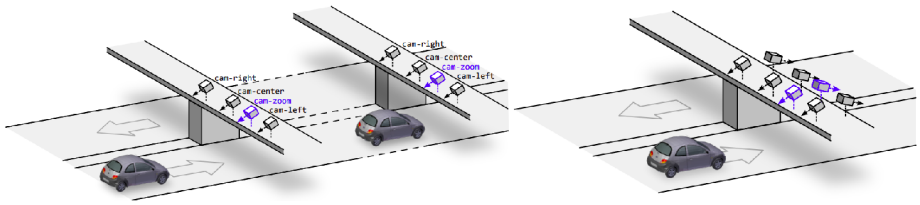
[4] https://unity.com/

[5] Evaluační systém pro úlohu re-identifikace vozidel v soutěži AI City Challenge na https://www.aicitychallenge.org/2021-evaluation-system/

4.4 CarsReId74k

Špaňhel et al. ve své práci *Learning Feature Aggregation in Temporal Domain for Re-Identification* [24] představil dataset CarsReId74k.

Snímky byly pořízeny ze 4 kamer umístěné na mostě nad dálnicí a dalších 4 kamer na jiném mostě (viz levou stranu obrázku 4.6). Další část datasetu byla pořízena z kamer umístěných na jednom mostu na jeho obou stranách (viz pravou stranu obrázku 4.6).

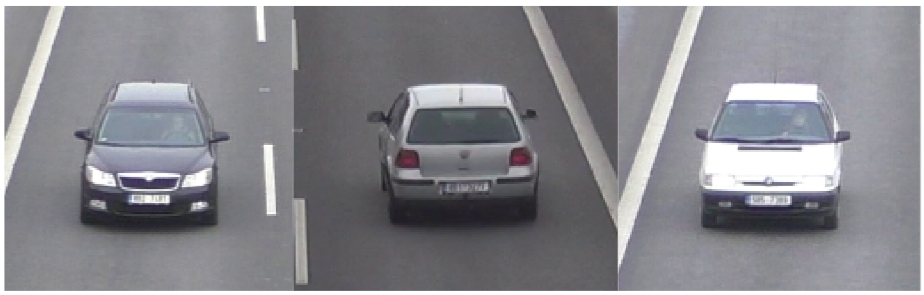


Obrázek 4.6: Nastavení kamer při pořizování datasetu CarsReId74k. Obrázek převzat z [24].

Celkem bylo vytvořeno 3 242 713 snímků vozidel zachycujících přes 17 000 unikátních identit. Data byla sbírána v 11 relacích na odlišných místech. Jedna z kamer byla vždy zaostřena tak, aby bylo možné využít automatického rozpoznávání poznávacích značek, které sloužily k vytvoření ground-truth anotací. Ostatní kamery byly namířené tak, aby zabíraly projíždějící vozidla zprava/zleva a zepředu/zezadu.

Trénovací část datové sady je tvořena 1 469 494 snímky. Testovací část obsahuje 1 467 680 snímků. Zbytek je validační část. Pro každou část byly shromážděny všechny páry trackletů, které patří stejnému vozidlu (označeny jako *query* a *positive*). Query a positive tracklety jsou vždy z různých videí. Positive párů je 277 236. Jako *negative* páry (query a negative) se použily všechny ostatní tracklety ve stejném videu jako positive tracklet. Průměrně je na jeden positive pár 1 283 negative trackletů. Pro evaluaci testovací sady je doporučeno využít přístupu image-to-tracklet.

Ukázka snímků z CarsReId74k je na obrázku 4.7.



Obrázek 4.7: Ukázka datasetu CarsReId74k.

4.5 Celkové porovnání datasetů

Kompletní statistiku a porovnání použitých datasetů nabízí tabulka 4.1. Dataset CarsReId74k obsahuje zdaleka nejvíce snímků vozidel. Nejvíce unikátních vozidel nabízí dataset VehicleID, který je ovšem omezen jen na přední a zadní pohledy. V porovnání s VehicleID

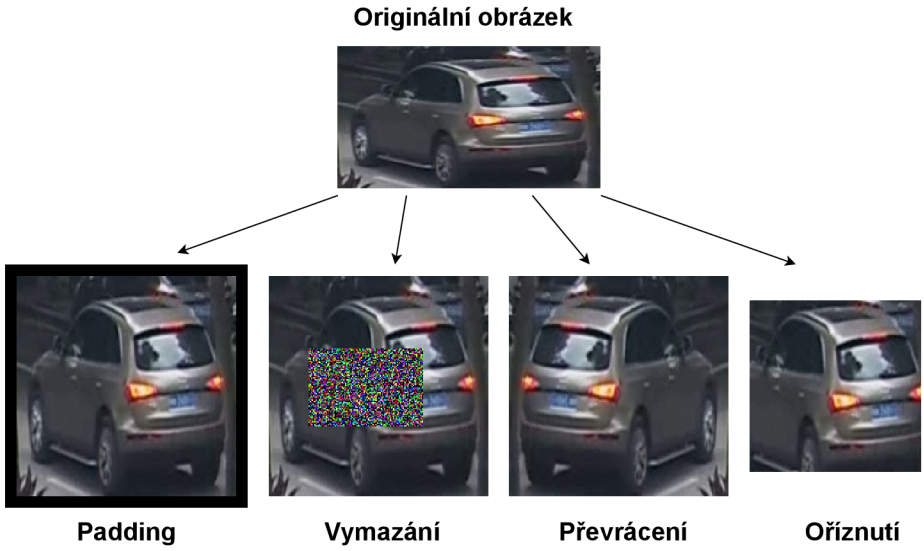
má CarsReId74k méně unikátních vozidel, ale mnohem více snímků, jelikož vozidla byla snímána z více pohledů.

	VeRi-776	VehicleID	CityFlow	VehicleX	CarsReId74k
Unikátní vozidla	776	**26 328**	880	1 362	17 681
Počet snímků	51 035	221 567	85 058	192 150	**3 242 713**
Pohledy na vozidla	Různé	Přední/Zadní	Různé	Různé	Různé
Syntetická data	Ne	Ne	Ne	Ano	Ne

Tabulka 4.1: Porovnání použitých datasetů k re-identifikaci vozidel.

4.6 Datová augmentace

Pro umělé rozšíření datové sady při trénování modelů jsem využil datové augmentace. Na každý obrázek jsem před tím, než vstoupil do modelu, použil sadu různých transformací, které se náhodně provedli nebo neprovedly. Jedná se o horizontální převrácení, oříznutí, padding a náhodné vymazání části obrázku. Obrázek 4.8 znázorňuje uvedené transformace.



Obrázek 4.8: Transformace použité při trénování modelu.

Kapitola 5

Experimenty

V rámci experimentů jsem pracoval s modely TransReID a CMT. Experimenty s modelem TransReID se zaměřovaly na využití klíčových bodů u datasetu VeRi-776. Cílem bylo zjistit, zda se vision transformery dokážou správně natrénovat i s využitím jen konkrétních částí vozidel. Model CMT nikdy nebyl využit k re-identifikaci vozidel. Cílem tedy bylo zjistit, zda je tento model pro tuto úlohu vhodný, provést jeho důkladné otestovaní a zjištění ideálních hyperparametrů v rámci této úlohy.

Veškeré experimenty byly implementovány v jazyce Python3 s využitím frameworku PyTorch[1] pro snadné trénování neuronových sítí. Trénování a výpočty jsem akceleroval pomocí grafické karty GeForce RTX 3070 Ti s pamětí 8GB. V průběhu práce mi byla poskytnuta grafická karta GeForce RTX 3090 s velikostí paměti 24GB, kterou jsem nadále využíval.

Podobně jako u prací [7, 3, 11] neuvádím u datasetu VehicleID metriku mAP, protože při testování je pro jeden query vstup vždy pouze jedna shoda v gallery.

5.1 Evaluační metriky

Pro měření úspěšnosti modelů, řešící re-identifikaci objektů, lze využít celou řadu metrik. Jedny ze základních metrik jsou tzv. *precision* a *recall*. Výpočet těchto metrik je dán vztahem:

$$\begin{aligned} precision &= \frac{true\ positives}{true\ positives + false\ positives}, \\ recall &= \frac{true\ positives}{true\ positives + false\ negatives}, \end{aligned} \tag{5.1}$$

kde *true positive* je počet správně predikovaných pozitivních tříd (model správně re-identifikoval objekty), *true negative* je počet správně predikovaných negativních tříd (model správně rozpoznal odlišné identity objektů), *false positive* je počet nesprávně predikovaných pozitivních tříd (model označil objekty s odlišnou identitou za stejné) a *false negative* je počet nesprávně predikovaných negativních tříd (model špatně označil objekty za různé, i když měli stejnou identitu).

Pro re-identifikaci objektů jsou ovšem nejpoužívanější metriky *Mean Average Precision* (mAP) a *Cumulative matching characteristics* (CMC) [34]. Tyto metriky pracují se dvěma

[1] Oficiální stránky frameworku PyTorch https://pytorch.org/

množinami identit – *query* a *gallery*. Typicky se v množině gallery hledají výskyty identit z query množiny.

Cumulative matching characteristics

Pro vyhodnocení CMC je každý prvek z množiny query porovnán se všemi prvky z množiny gallery. Během porovnávání jsou prvky z množiny gallery seřazeny vzestupně do n-tice podle vzdáleností od query prvku. Za předpokladu, že v množině gallery je N prvků, je seřazená n-tice gallery dána jako $ranked = (r_1, r_2, \ldots, r_N)$. Pro rank k je pak CMC přesnost dána následovně:

$$Acc(k, ranked) = \begin{cases} 1, & ID_q \in (ID_{r1}, ID_{r2}, \ldots, ID_{rk}) \\ 0, & \text{jinak} \end{cases}, \tag{5.2}$$

kde ID_q je identita query prvku a ID_{r_i} je identita gallery prvku na pozici i v n-tici $ranked$. Výsledné CMC přes M query prvků je pak dáno jako:

$$CMC(M, k) = \frac{1}{M} \sum_{i=1}^{M} Acc(k, ranked_i), \tag{5.3}$$

kde $ranked_i$ je vzestupně seřazená n-tice gallery prvků podle vzdálenosti od query prvku i. Je zřejmé, že CMC vyjadřuje pravděpodobnost nalezení správného prvku v prvních k prvcích seřazené sekvence. Typicky se používá k=1, k=5, k=10 a k=20.

Aby CMC dávalo relevantní výsledky, musí být splněna podmínka, kdy se pouze jedna identita query prvku nachází v množině gallery [37]. V případě, kdy v množině gallery existuje vícero identit query prvku, může být CMC zavádějící, protože nebere v potaz recall. Na obrázku 5.1 je takováto situace znázorněna. Z toho důvodu se často při re-identifikaci využívá metrika mAP [38].

Mean avegare precision

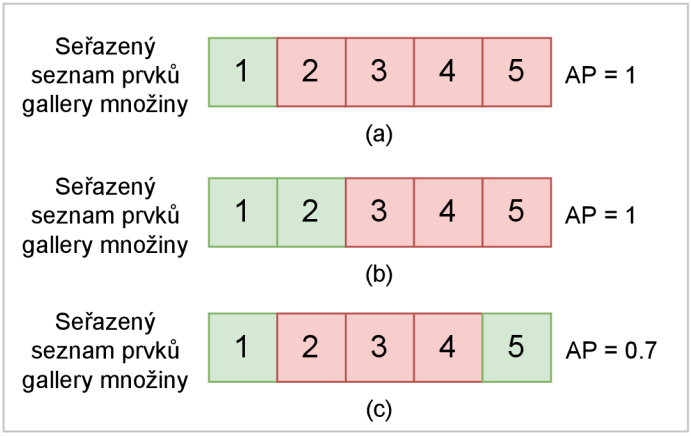
Pro spočítání mAP metriky se pro každý query prvek q spočítá *average precision* (AP) následovně [34]:

$$AP(q_i) = \frac{\sum_{k=1}^{N} P(k) \times G(k)}{N_{gt}}, \tag{5.4}$$

kde N je počet prvků v množině gallery, N_{gt} je počet identit v množině gallery shodných s identitou query prvku, $P(k)$ je precision (viz vztah 5.1) v k dlouhé seřazené n-tici a $G(k)$ je rovno 1 pokud se identita k-tého prvku shoduje s identitou query prvku, jinak je rovno 0. Mean average precision pak vyjadřuje následující vztah [34]:

$$mAP = \frac{\sum_{q=1}^{Q} AP(q)}{Q}, \tag{5.5}$$

kde Q je počet query prvků. Rozdíl mezi CMC a mAP je znázorněn na obrázku 5.1.



Obrázek 5.1: Rozdíl mezi metrikami CMC a AP. V zelených čtvercích jsou pozitivní shody s query prvkem a v červených negativní. Pro všechny tři seřazené seznamy (a), (b) i (c) je CMC=1. Pro (a) je AP=1, pro (b) je AP=1 a pro (c) je AP=0,7. Hodnota mAP je rovna 0,9. Obrázek převzat z [37].

5.2 TransReID

Při prvních experimentech jsem se věnoval modelu TransReID (viz podkapitolu 3.2). Autoři zveřejnili zdrojový kód k natrénování a testování modelu TransReID na svém githubu[2]. Jejich kód jsem převzal a upravil podle svých potřeb. Jako backbone modelu jsem zvolil předtrénovaný model ViT[3] (viz podkapitolu 2.2) na datasetu *ImageNet*[4].

Při prvním experimentu jsem model natrénoval na stejných datasetech jako autoři, tedy VeRi-776, VehicleID a CityFlow. V původní implementaci měly vstupní obrázky velikost 256×256 a velikost batch byla nastavena na 96 obrázků. Z důvodů nedostatečné velikosti paměti na grafické kartě (8GB), jsem byl nucen snížit velikost vstupu na 128×128 a velikost batch na 32. Na datasetu VeRi-776 a VehicleID trénování probíhalo 120 epoch. Loss funkce byla složena z ID loss a triplet loss. Vývoj loss hodnoty během trénování na obou datasetech je na obrázku 5.2. Za optimalizér jsem zvolil SGD. Learning rate prvních 5 epoch lineárně rostla k hodnotě 0,01 a poté klesala s kosinovým trendem. Dosažené výsledky jsou v tabulce 5.1.

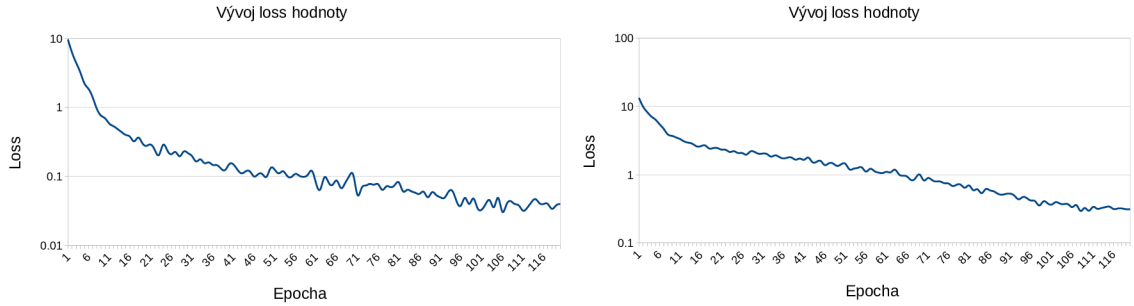
Model	Dataset	mAP	Rank1	Rank5
TransReID (orig.)	VeRi-776	**0,805**	**96,8**	—
TransReID	VeRi-776	0,767	96,3	**98,0**
TransReID (orig.)	VehicleID-800	—	**85,2**	**97,5**
TransReID	VehicleID-800	—	77,6	94,4

Tabulka 5.1: Výsledky mnou natrénovaného modelu TransReID na datasetech VeRi-776 a VehicleID v porovnání s výsledky originálního TransReID.

[2]Zdrojový kód modelu TransReID a upraveného modelu pro dataset CityFlow dostupný na https://github.com/damo-cv/TransReID, respektive https://github.com/michuanhaohao/AICITY2021_Track2_DMT

[3]Váhy modelu ViT volně ke stažení na https://github.com/rwightman/pytorch-image-models/releases/download/v0.1-vitjx/jx_vit_base_p16_224-80ecf9dd.pth

[4]https://www.image-net.org/



(a) VeRi-776 (b) VehicleID

Obrázek 5.2: Vývoj loss hodnoty při trénování modelu TransReID na datasetech VeRi-776 a VehicleID. Osy loss jsou v logaritmickém měřítku.

Oproti výsledkům dosaženými původními autory došlo ke zhoršení. Na datasetu VeRi-776 byl můj výsledek o 0,038 horší na metrice mAP a na datasetu VehicleID došlo ke zhoršení o 7,6% na metrice Rank1. Toto zhoršení lze vysvětlit částečně zmenšením obrazu, kdy dojde ke ztrátě informací, ale především sníženou velikostí batch. V triplet loss se používá online batch mining, a snížení z 96 vzorků na 32 mělo velký efekt na úspěšnost.

Trénování modelu na datasetu CityFlow probíhalo lehce odlišně oproti trénování na ostatních datasetech, jelikož bylo rozděleno na dvě fáze. Důvodem této změny byl fakt, že v testovací sadě datasetu CityFlow se objevuje úplně nová scéna trati, která se v trénovací sadě neobjevuje. První fáze probíhala totožně jako na předchozích datasetech. Natrénoval jsem dvě varianty. První varianta využila pouze dataset CityFlow a druhá využila kombinaci datasetů CityFlow a VehicleX. Model se trénoval 40 epoch. Vývoj loss hodnoty při trénování na kombinaci CiyFlow a VehicleX je znázorněn na obrázku 5.3. Ostatní parametry trénování se nezměnily.

Ve druhé fázi se uplatnila *unsupervised domain-adaptive* (UDA) metoda. UDA vygenerovala *pseudo-labely* pro testovací data, které se pak použily k doladění modelu. Pro vygenerování pseudo-labelů je potřeba pro všechny testovací obrázky nejdříve vygenerovat příznaky, které jsou pak vstupem do shlukovacího algoritmu, který vygeneruje pseudo-labely [17].

Pro obrázek I, který pořídila kamera C v *trackletu* T, je jeho globální příznak označen jako g_i (ten lze získat z natrénovaného modelu v první fázi). Dále je příznak, který je průměrem všech příznaků z obrázků zachycených kamerou C, označen jako $\bar{g}_C$. Ke snížení odchylky mezi obrázky se stejnou identitou vozidla zachycenými z různých kamer, je tzv. *single-frame* příznak f_I obrázku I spočítán následovně:

$$f_I = g_i - \alpha \bar{g}_C, \tag{5.6}$$

kde α je balanční váha mezi g_i a $\bar{g}_C$ [17].

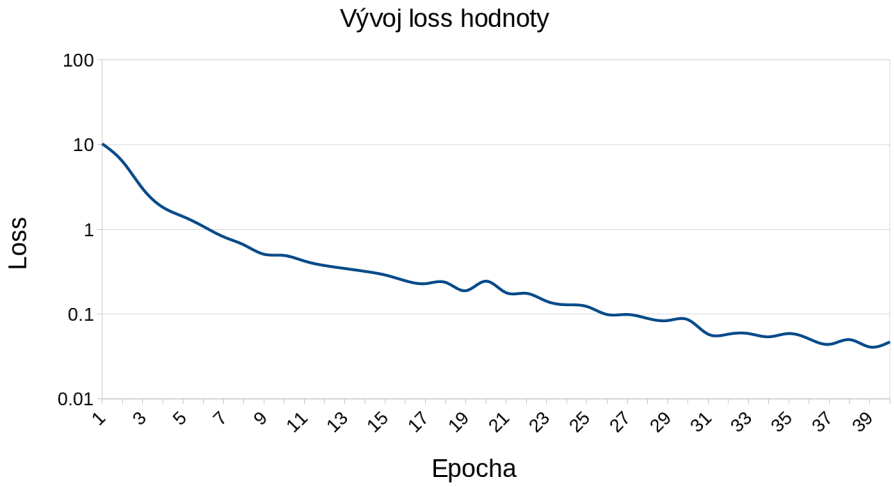
Protože informace o ID trackletu je v datasetu CityFlow známá pro každý obrázek, je možné zprůměrovat všechny příznaky trackletu T pro získání tracklet příznaku t_i pro obrázek I.

Výsledný příznak pro obrázek I je vypočítán tedy následovně [17]:

$$\hat{f}_I = \beta f_I + (1 - \beta) t_i, \tag{5.7}$$

kde β je balanční váha mezi single-frame a traťovým příznakem. $\hat{f}_I$ je pak vstupem do shlukovacího algoritmu pro vygenerování pseudo-labelů. Autoři zvolili DBSCAN[5] algoritmus, který jsem využil také. Model z první fáze je pak doladěn pomocí vygenerovaných pseudo-labelů.

Model jsem ve druhé fázi trénoval 3 epochy a β jsem nastavil na 0,3 a α na 0,0005. Maximální vzdálenost mezi vozidly pro vytvoření shluku algoritmem DBSCAN byla nastavena na 0,55. Trénování probíhalo pouze na datasetu CityFlow. Také jsem odstranil JPM a SIE větve, jelikož dataset CityFlow neposkytuje potřebné informace pro SIE modul. Dle autorů větev JPM nepřináší žádné zlepšení na datasetu CityFlow [17]. Při evaluaci jsem využil přístup image-to-track. Také jsem využil při postprocessingu metody rerankingu. Dosažené výsledky z obou fází znázorňuje tabulka 5.2.



Obrázek 5.3: Vývoj loss hodnoty při trénování modelu TransReID na datasetu CityFlow při fázi 1. Osa Loss je v logaritmickém měřítku.

Model	Dataset	Fáze 1		Fáze 2	
		mAP	Rank1	mAP	Rank1
TransReID (orig.)	CityFlow + VehicleX	—	—	**0,565**	**69,7**
TransReID	CityFlow	0,365	50,8	0,401	56,1
TransReID	CityFlow + VehicleX	**0,376**	**52,9**	0,423	56,9

Tabulka 5.2: Výsledky mnou natrénovaného upraveného modelu TransReID pro dataset CityFlow. Byly natrénovány dvě varianty. První pouze na datasetu CityFlow a druhá na kombinaci datasetů CityFlow a VehicleX. V první fázi se model učil na uvedeném datasetu. Ve druhé fázi proběhlo UDA trénování. Při evaluaci byl využit přístup image-to-track a reranking.

Dále jsem model natrénoval na datasetu CarsReId74k. Modelu jsem odebral SIE větev. Větev JPM jsem zachoval. Trénování proběhlo v 50 epochách. Při evaluaci jsem využil doporučovaného přístupu image-to-track (viz podkapitolu 4.4). Výsledná reprezentace trackletu

[5]Popis DBSCAN dostupný na https://en.wikipedia.org/wiki/DBSCAN

byla vytvořena z průměru jednotlivých příznaků obrázků v trackletu. Výsledky zobrazuje tabulka 5.3.

Model	mAP	Rank1	Rank5
TransReID	0,651	54,7	77,8

Tabulka 5.3: Výsledky modelu TransReID na datasetu CarsReId74k. Model měl přítomnou pouze větev JPM. Pro evaluaci byl využit přístup image-to-track.

Klíčové body

Při dalších experimentech s modelem TransReID jsem chtěl zjistit, zda vision transformery dokáží úspěšně rozlišit identitu vozidla s využitím jen určitých oblastí vozidla na obrázku. Tím by se odstranil šum, který by mohl být v okolí vozidla a zároveň by se tak modelu pomohlo zaměřit se jen na ty nejdůležitější oblasti vozidla. Druhým efektem tohoto přístupu je snížení výpočetní náročnosti, jelikož model pracuje s méně vstupy. K tomuto úkolu se hodí dataset VeRi-776, který obsahuje anotované klíčové body na vozidle (viz podkapitolu 4.1). Z tohoto důvodu byly následující experimenty provedeny pouze na tomto datasetu.

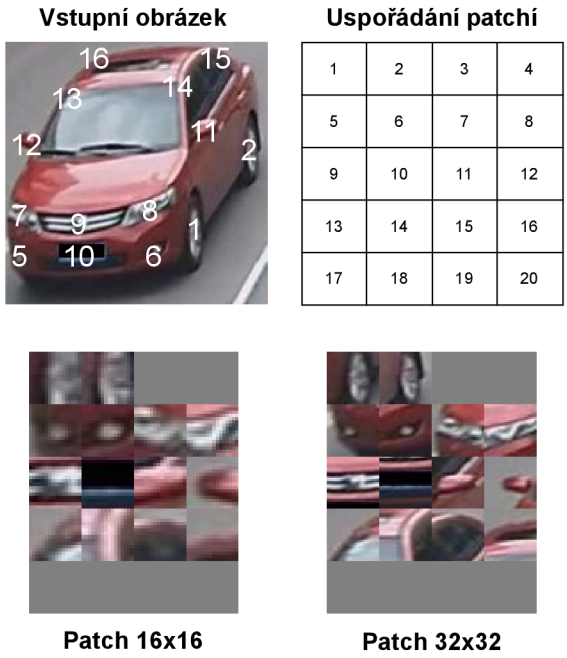
V originální implementaci modelu TransReID je vstupem celý obrázek, který je konvoluční vrstvou rozdělen na patche. Tato vrstva se skládá z 768 konvolučních jader o rozměrech 16×16 a velikosti kroku 12. V případě vstupního obrázku o rozměru 192×192 je vyextrahováno celkem 768 map příznaků o rozměru 15×15, což dělá 225 příznaků na mapu. Každá mapa se pak narovná, čímž vznikne vstup o rozměru 768×225. Z tohoto výstupu se pak pomocí transpozice vytvoří výsledný embedding o rozměru 225×768. Tento embedding pak pokračuje dál do modelu.

V případě řešení s klíčovými body bylo nejprve nutné sestavit vstupní obrázek. Jako první se na originální obrázek aplikovaly datové augmentace (viz podkapitolu 4.6). Poté pro každý viditelný klíčový bod na vozidle byla vyextrahována patch o velikosti 16×16, kde středem této patche byl právě klíčový bod. Pro body, které nebyly viditelné, byla vytvořená patch vyplněna předem definovanou hodnotou. Abych do trénovacích dat přinesl větší variabilitu, posouval jsem každý klíčový bod náhodně o 3 pixely doleva/doprava a nahoru/dolů. Při tomto procesu bylo celkem vytvořeno 20 patchí, které jsem sestavil do jednoho výsledného obrázku, který měl 4 patche na řádek a 5 patchí na sloupec. Obrázek 5.4 znázorňuje tento proces tvoření patchí.

Obrázek sestavený z patchí měl rozměry 64×80. Konvoluční vrstvě jsem upravil velikost kroku na hodnotu 16, protože jednotlivé patche na sebe nenavazovaly. Celkem bylo vyextrahováno 768 map příznaků o rozměru 4×5, a tedy vznikl embedding o rozměru 20×768.

První experiment měl za úkol zjistit, zda větve SIE a JPM přinesou zlepšení i v tomto nastavení. Natrénoval jsem tedy TransReID ve všech kombinací. Trénování probíhalo 120 epoch. Jelikož měly vstupní obrázky výrazně menší velikost než při klasickém trénování, zvětšil jsem velikost batch na 128. Za optimalizér jsem zvolil SGD. Tabulka 5.4 obsahuje dosažené výsledky.

Při použití obou větví dohromady jsem dosáhl na metrice mAP hodnoty 0,522, což je téměř o 0,06 více než při nepoužití žádné větve. Každá větev zlepšila výsledky a jejich kombinace dohromady dosáhla výsledků nejlepších. V následujících experimentech tedy byly obě větve přítomny.



Obrázek 5.4: Vstupní obrázek rozdělen na patche podle klíčových bodů. Indexy bodů odpovídají anotacím z obrázku 4.2. Vyzkoušel jsem extrahovat patche o rozměru 16×16 a 32×32. Pokud klíčový bod není na obrázku viditelný, je patch vyplněna předem definovanou hodnotou.

Model	SIE	JPM	mAP	Rank1	Rank5
TransReID	—	—	0,468	79,5	90,6
TransReID	✓	—	0,515	83,5	92,5
TransReID	—	✓	0,498	81,9	91,6
TransReID	✓	✓	**0,522**	**83,6**	**93,1**

Tabulka 5.4: Výsledky modelu TransReID s využitím klíčových bodů a větví JPM a SIE.

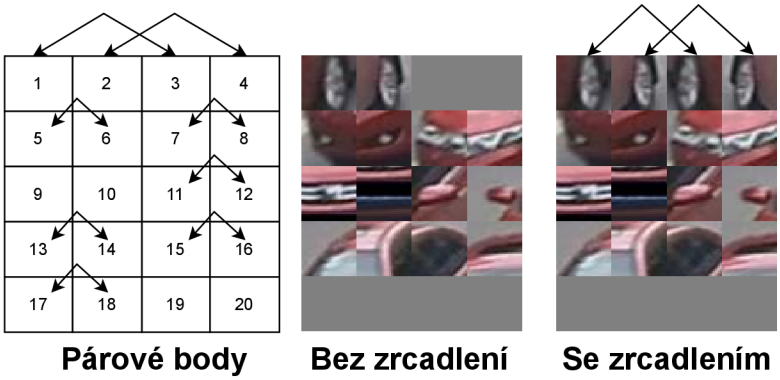
Dále jsem chtěl zjistit, zda a případně, jak moc záleží na hodnotě, kterou vyplňuji patche z klíčových bodů, které nejsou vidět. V průměru je takových bodů 42%, a proto je zapotřebí, aby co nejméně ovlivňovaly model. Vyzkoušel jsem hodnoty 0, 1 a 0,5. Samotné trénování probíhalo identicky jako při předchozím experimentu. Dosažené výsledky shrnuje tabulka 5.5.

Model	Hodnota	mAP	Rank1	Rank5
TransReID	0	0,518	83,3	93,0
TransReID	0,5	**0,522**	**83,6**	**93,1**
TransReID	1	0,521	82,5	92,2

Tabulka 5.5: Výsledky modelu TransReID při různé volbě hodnoty, kterou se vyplňují patche, které nejsou viditelné.

Ačkoli nejlepších výsledků jsem dosáhl s použitím hodnoty 0,5, tak při použití jiných hodnot nebyly výsledky o moc horší. Vliv této hodnoty je tedy zanedbatelný. Při dalších experimentech jsem se rozhodl nadále používat hodnotu 0,5.

Jak již bylo zmíněno, klíčových bodů, které nejsou na vozidle vidět, je v průměru 42%. Tedy téměř půlka vstupního obrázku je tvořena neinformativními hodnotami. Z 20 klíčových bodů lze 16 označit za *párové*. Párové body jsou dvojice bodů, které při horizontálním otočení budou vypadat stejně jako jejich protějšek. V případě, kdy klíčový bod není viditelný, ale jeho párový bod ano, vytvoří se pro tento neviditelný bod patch z horizontálně otočené patche, která bylo vytvořena z viditelného bodu. Příkladem budiž vozidlo zachycené z levého boku. Pro jeho pravou přední pneumatiku lze využít horizontálně otočená levá přední pneumatika. Při tomto tvoření patchí mají pak vstupní obrázky v průměru 33% neinformativních hodnot. Tuto techniku jsem označil jako *zrcadlení*. Obrázek 5.5 znázorňuje párové klíčové body a výsledný obrázek po aplikaci zrcadlení. Trénování se zrcadlením probíhalo se stejnými parametry jako předchozí experiment. Tabulka 5.6 ukazuje dosažené výsledky.



Obrázek 5.5: Technika zrcadlení klíčových bodů. Indexy bodů odpovídají anotacím z obrázku 4.2.

Model	Zrcadlení	mAP	Rank1	Rank5
TransReID	—	0,522	83,6	**93,1**
TransReID	✓	**0,536**	**83,7**	92,7

Tabulka 5.6: Výsledky modelu TransReID s využitím metody zrcadlení klíčových bodů.

Zrcadlení zlepšilo výsledek na metrice mAP o 0,014. Na ostatních metrikách nedošlo k výrazné změně. Proto jsem v dalších experimentech model trénoval bez i se zrcadlením, abych více ověřil přínos této techniky.

Z konvoluční vrstvy se 768 konvolučními jádry lze získat ze vstupního obrázku, který má rozměr 64×80, embedding o rozměru 20×768. V porovnání s originální implementací, kde z konvoluční vrstvy, v případě vstupního obrázku o rozměru 192×192, šlo vygenerovat embedding o rozměru 225×768, došlo k výrazné redukci množství vstupních dat. To mohlo mít za následek propad v úspěšnosti v porovnání s implementací bez klíčových bodů. Pro zvýšení množství vstupních dat jsem jako první vyzkoušel zvětšit rozměry patchí na 32×32 z původních 16×16 (viz obrázek 5.4). Z embeddingu o rozměru 20×768 se tak stal embedding o rozměru 80×768. Rozměr vstupního obrázku se zvýšil na 128×160. Větší

patche také dokáží zachytí větší oblast kolem klíčového bodu, takže model bude mít více informací k extrahování.

Trénování s větším rozměrem patchí probíhalo ve stejném nastavení jako v předchozím experimentu. Tabulka 5.7 shrnuje dosažené výsledky.

Model	Velikost patche	Zrcadlení	mAP	Rank1	Rank5
TransReID	16x16	✓	0,536	83,7	92,7
TransReID	32x32	—	0,615	88,3	95,1
TransReID	32x32	✓	**0,622**	**88,4**	**95,4**

Tabulka 5.7: Výsledky modelu TransReID s různou velikostí patche.

Zvětšení rozměru patchí přineslo výrazné zlepšení. Na metrice mAP byl rozdíl téměř 0,08 a po přidání zrcadlení dokonce 0,086.

Další metoda, kterou jsem vyzkoušel pro zvýšení počtu vstupních dat, bylo přidání k sestavenému obrázku jeho horizontálně otočenou variantu. Z konvoluční vrstvy se tak vygeneroval embedding o rozměru 160×768. Rozměr vstupního obrázku se zvýšil na 256×160.

Trénování probíhalo ve stejném nastavení jako v předchozím experimentu. Zrcadlení patchí jsem zachoval. Tabulka 5.8 shrnuje dosažené výsledky.

Model	Horiz. otočení obrázku	mAP	Rank1	Rank5
TransReID	—	0,622	88,4	**95,4**
TransReID	✓	**0,637**	**89,4**	**95,4**

Tabulka 5.8: Výsledky modelu TransReID s použitím vstupního obrázku společně s jeho horizontálně otočenou variantou.

Přidání ke vstupnímu obrázku jeho horizontálně otočenou variantu se prokázalo jako užitečné. Zlepšení na metrice mAP bylo o 0,015. Nežádoucím efektem bylo výrazné zvýšení vstupních dat, a tedy se zvýšila i výpočetní náročnost. Při tomto přístupu už se ovšem nevyplatí využívat klíčové body, jelikož rozměr vstupního obrázku je tak velký, že není důvod nevyužít celého obrázku s vozidlem.

Posledním experimentem bylo přidání rerankingu. Pro tento test jsem využil model natrénovaný z předchozího experimentu. Tabulka 5.9 ukazuje dosažené výsledky. Reranking výrazně zvýšil úspěšnost modelu, a to o 0,111 na metrice mAP.

Model	Reranking	mAP	Rank1	Rank5
TransReID	—	0,637	89,4	**95,4**
TransReID	✓	**0,748**	**91,1**	94,9

Tabulka 5.9: Výsledky modelu TransReID s využitím rerankingu.

5.3 CMT

Další experimenty jsem věnoval modelům CMT (viz podkapitolu 3.3). První experiment měl za úkol zjistit, zda model ve své původní podobě zvládne dosáhnout uspokojivých výsledků na úloze re-identifikace. Výstup klasifikátoru modelu je vektor s jednotlivými pravděpodobnostmi, které určují příslušnost pro danou třídu. I takové nastavení lze využít pro natrénování re-identifikace vozidel. Každý prvek ve výstupním vektoru z klasifikátoru bude představovat jednu identitu vozidla. Při každém experimentu byl trénovaný model předtrénován na datasetu ImageNet.

Implementaci samotného modelu autoři zveřejnili na svém githubu[6]. Jejich implementaci jsem převzal a zasadil do svého frameworku pro trénování a vyhodnocování neuronových sítí.

Experimenty s neupraveným modelem CMT

První experimenty jsem provedl na modelu CMT-ti z rodiny CMT. Za loss funkci jsem zvolil ID loss. Velikost batch jsem nastavil na 128. Model se trénoval po 300 epoch. Vstupem ID loss byl výstup poslední vrstvy modelu, tedy klasifikátoru. Při vyhodnocování se jako reprezentace vstupního obrázku bral embedding z předposlední vrstvy modelu, tedy po závěrečné konvoluci a average poolingu. Vstupní obrázek měl rozměry 160×160.

Stejný experiment jsem provedl také s větším modelem CMT-xs. Nastavení trénování bylo stejné jako s modelem CMT-ti. Jediný rozdíl byl v počtu trénovacích epoch, který byl nastaven na 80 a rozměry vstupního obrázku byly 192×192.

V obou experimentech jsem použil AdamW optimalizér[7]. Pro upravování learning rate během trénování jsem využil algoritmus StepLR, kde každou desátou epochu se learning rate snížila o polovinu.

Tabulka 5.10 ukazuje výsledky experimentu na CMT-ti a CMT-xs. Obrázek 5.6 znázorňuje vývoj loss hodnoty a learning rate pro CMT-ti i CMT-xs při trénování na datasetu VeRi-776. Jak je vidět, model CMT-ti se natrénoval brzo a 300 epoch bylo více než dost.

CMT-ti i CMT-xs modely, které nebyly upraveny na úlohu re-identifikace vozidel, dosáhly zjevně dobrých výsledků. Model CMT-xs, který je větší než CMT-ti, dosáhl na každém z datasetů lepších výsledků. Na datasetu VeRi-776 dosáhl model CMT-xs při Rank1 92,1%, což se již velice blíží nejpřesnějším modelům pro re-identifikaci vozidel. Z výsledků experimentů tedy lze předpokládat, že upraví-li se CMT vhodně, má potenciál dosáhnout state-of-the-art úspěšnosti. Jako první jsem se tedy zaměřil na správné nastavení loss funkce.

Experimenty s triplet loss

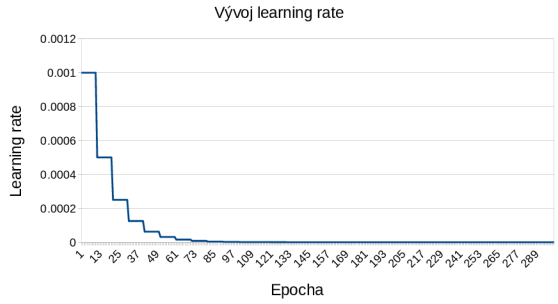
Při experimentech s loss funkcí jsem vyzkoušel dvě možnosti. Samotnou triplet loss a kombinaci ID a triplet loss. Tentokrát jsem experimentoval pouze na modelu CMT-xs. Pro testování samotné triplet loss bylo potřeba zjistit vhodnou margin hodnotu (viz rovnici 2.7). Otestoval jsem hodnoty od 0,2 po 1,0 s přírůstkem 0,2 na datasetu VeRi-776. Jako u předchozích experimentů, i zde jsem použil AdamW optimalizér a StepLR. Velikost batch byla nastavena na 128 a trénovalo se po 80 epoch. Pro získání výstupního embeddingu jsem využil konvoluční vrstvu, která měla 512 konvolučních jader o velikosti 1×1 a velikost kroku 1, následovanou batch normalizací a 2D average poolingem. Tento embedding pak byl vstupem do triplet loss. Výstup klasifikátoru se v tomto případě nebral v potaz.

[6] Implementace modelu CMT dostupná na https://github.com/ggjy/CMT.pytorch.

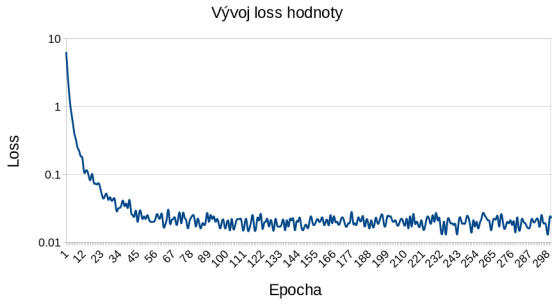
[7] AdamW optimalizér https://pytorch.org/docs/stable/generated/torch.optim.AdamW.html

Dataset	Model	mAP	Rank1	Rank5
VeRi-776	CMT-ti	0,592	88,8	94,3
VeRi-776	CMT-xs	**0,610**	**92,1**	**96,7**
VehicleID-800	CMT-ti	—	61,9	70,5
VehicleID-800	CMT-xs	—	**69,3**	**78,6**
VehicleID-1600	CMT-ti	—	60,5	69,3
VehicleID-1600	CMT-xs	—	**67,8**	**75,3**
VehicleID-2400	CMT-ti	—	57,4	66,3
VehicleID-2400	CMT-xs	—	**66,2**	**73,6**

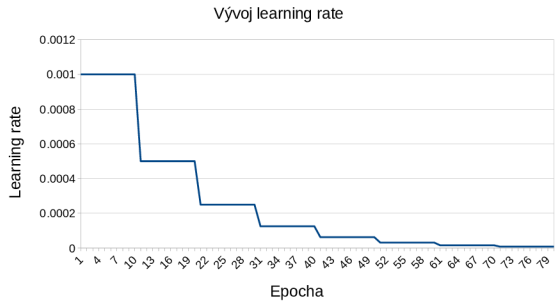
Tabulka 5.10: Výsledky modelu CMT-ti i CMT-xs neuzpůsobených k re-identifikaci vozidel. Jako výstup se využíval embedding z předposlední vrstvy modelů. Trénováno s použitím ID loss.



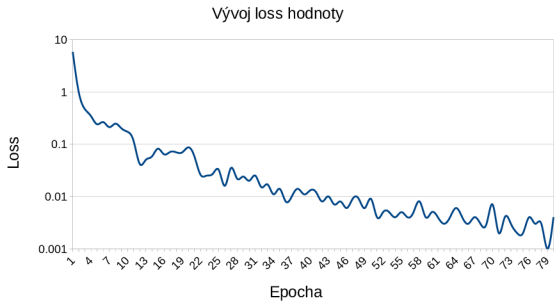
(a) CMT-ti



(b) CMT-ti



(c) CMT-xs



(d) CMT-xs

Obrázek 5.6: Vývoj learning rate (a) a loss hodnoty (b) při trénování neupraveného modelu CMT-ti. To stejné pro CMT-xs (c) a (d). Trénováno na datasetu VeRi-776. Osy s loss hodnotou jsou v logaritmickém měřítku.

Z experimentu na VeRi-776 vyšla nejlépe hodnota 0,2. Druhé nejlepší nastavení marginu bylo 0,4. Proto jsem tyto dvě hodnoty dále otestoval na datasetu VehicleID. Trénování proběhlo v 80 epochách. Velikost batch zůstala na 128. Všechny dosažené výsledky shrnuje tabulka 5.11.

Jak je vidět, nejlepších výsledků bylo dosaženo s hodnotou margin 0,2 na obou datasetech. Proto při dalších experimentech měla triplet loss vždy margin nastaven na tuto

Dataset	Margin	mAP	Rank1	Rank5
VeRi-776	0,2	**0,730**	**94,0**	**97,3**
VeRi-776	0,4	0,729	93,4	97,0
VeRi-776	0,6	0,717	92,2	96,8
VeRi-776	0,8	0,724	92,2	97,1
VeRi-776	1,0	0,728	93,1	97,0
VehicleID-800	0,2	—	81,0	**97,2**
VehicleID-800	0,4	—	**81,1**	96,1
VehicleID-1600	0,2	—	**79,9**	**95,3**
VehicleID-1600	0,4	—	75,1	93,5
VehicleID-2400	0,2	—	**74,3**	**92,1**
VehicleID-2400	0,4	—	70,3	90,8

Tabulka 5.11: Výsledky modelu CMT-xs s využitím triplet loss a různých nastavení margin hodnoty na datasetech VeRi-776 a VehicleID.

hodnotu. Samotná změna z ID loss na triplet loss velice znatelně zlepšila výsledky, nehledě na hodnotu margin. Na obou datasetech se úspěšnost zlepšila zhruba o 10% na všech metrikách.

Dalším experimentem byl test kombinace ID loss spolu s triplet loss. Výsledná loss funkce lze formálně vyjádřit jako:

$$L = L_{ID} + L_{Triplet}, \tag{5.8}$$

kde L_{ID} je cross-entropy loss funkce a $L_{Triplet}$ je triplet loss. Vstupem triplet loss byl stejný embedding jako při předchozím experimentu. Triplet loss měla nastaven margin na 0,2. Trénování probíhalo celkem 80 epoch na modelu CMT-xs. Ostatní parametry trénování zůstaly stejné jako při předchozím trénování. Výsledky experimentu shrnuje tabulka 5.12.

Dataset	Loss funkce	mAP	Rank1	Rank5
VeRi-776	triplet	**0,730**	94,0	97,3
VeRi-776	ID + triplet	0,728	**95,2**	**98,0**
VehicleID-800	triplet	—	**81,0**	**97,2**
VehicleID-800	ID + triplet	—	80,2	94,9
VehicleID-1600	triplet	—	**79,9**	**95,3**
VehicleID-1600	ID + triplet	—	78,0	93,1
VehicleID-2400	triplet	—	74,3	**92,1**
VehicleID-2400	ID + triplet	—	**75,7**	90,0

Tabulka 5.12: Výsledky modelu CMT-xs s využitím triplet loss a kombinací ID a triplet loss. Výsledky modelu s použitím jen triplet loss pochází z tabulky 5.11 při použití margin hodnoty 0,2.

Na datasetu VehicleID kombinace ID loss a triplet loss dosáhla o pár procent horších výsledků na všech metrikách (až na Rank1 při VehicleID-2400) než při použití samotné triplet loss. Na datasetu VeRi-776 kombinace loss funkcí dosáhla lepších výsledků při Rank1

a Rank5, ale horších na metrice mAP. Celkově se dá označit trénování se samotnou triplet loss jako úspěšnější. Další experimenty byly tedy provedeny s tímto nastavením loss funkce.

Vliv velikosti batch

Pro triplet loss využívám online triplet mining. To znamená, že trojice vstupů pro triplet loss se vybírá z trénovací batche. Cílem tohoto experimentu bylo zjištění, jak moc velikost batch ovlivňuje úspěšnost modelu. Výsledky pro různé nastavení batch velikosti uvádí tabulka 5.13.

Dataset	Batch	mAP	Rank1	Rank5
VeRi-776	32	0,460	73,5	87,5
VeRi-776	64	0,696	90,8	96,5
VeRi-776	128	**0,730**	**94,0**	**97,3**
VehicleID-2400	32	—	41,9	70,6
VehicleID-2400	64	—	56,2	82,0
VehicleID-2400	128	—	**74,3**	**92,1**

Tabulka 5.13: Vliv velikosti batch na výsledky modelu CMT-xs.

Z výsledků lze vypozorovat, že velikost batch má výrazný vliv na úspěšnost. Snížení ze 128 na 64 přineslo zhoršení téměř 20% na metrice Rank1 u datasetu VehicleID. Stejně tak na datasetu VeRi-776 se úspěšnost výrazně zhoršila. V dalších experimentech jsem nechal batch velikost nastavenou na hodnotě 128.

Metrika podobnosti embeddingů

Další experiment se věnoval zvolení vhodné metriky k určení podobnosti mezi výslednými embeddingy. Vyzkoušel jsem dva přístupy. Prvním byla euklidovská vzdálenost, která byla používaná v předchozích experimentech. Definovat lze následovně:

$$d_{Euc}(x, y) = \sqrt{\sum_{i=1}^{n}(x_i - y_i)^2}, \tag{5.9}$$

kde x a y jsou vektory, mezi kterými se má vzdálenost spočítat a n je počet složek ve vektoru.

Druhou možností je kosinová podobnost, kterou lze definovat následovně:

$$d_{Cos}(x, y) = \frac{\sum_{i=1}^{n} x_i y_i}{\sqrt{\sum_{i=1}^{n} x_i^2}\sqrt{\sum_{i=1}^{n} y_i^2}}, \tag{5.10}$$

kde x a y jsou vektory, mezi kterými se má podobnost spočítat a n je počet složek ve vektoru. Výsledkem kosinové podobnosti je číslo v rozmezí od -1 do 1, kde -1 znamená, že vektory jsou opačné, 0 znamená, že jsou ortogonální a 1 znamená, že jsou stejné.

Natrénoval jsem tedy model CMT-xs s použitím kosinové podobnosti a porovnal s euklidovskou vzdáleností, která se využívala v předchozích experimentech. Parametry trénování se jinak nezměnily. Tabulka 5.14 shrnuje dosažené výsledky.

Dataset	Metrika	mAP	Rank1	Rank5
VeRi-776	Eukl. vzdálenost	**0,730**	**94,0**	**97,3**
VeRi-776	Kosinová podobnost	0,682	92,5	95,9
VehicleID-800	Eukl. vzdálenost	—	**81,0**	**97,2**
VehicleID-800	Kosinová podobnost	—	80,7	95,6
VehicleID-1600	Eukl. vzdálenost	—	**79,9**	**95,3**
VehicleID-1600	Kosinová podobnost	—	77,5	93,0
VehicleID-2400	Eukl. vzdálenost	—	**74,3**	**92,1**
VehicleID-2400	Kosinová podobnost	—	73,6	90,4

Tabulka 5.14: Výsledky modelu CMT-xs s použitím různých metrik na vyhodnocení podobnosti vektorů. Výsledky modelu s využitím euklidovské vzdálenosti pochází z tabulky 5.12.

Na obou datasetech jsem dosáhl lepších výsledků s použitím metriky euklidovské vzdálenosti. Například na datasetu VeRi-776 byl rozdíl u mAP téměř 0,05. V dalších experimentech jsem tedy pokračoval s používáním euklidovské vzdálenosti.

Tvorba výstupního embeddingu

Doposud jsem tvořil výstupní embedding z konvoluční vrstvy, která měla 512 konvolučních jader o velikosti 1×1 a velikost kroku 1 následovanou batch normalizací a 2D average poolingem. V tomto experimentu jsem se zaměřil na samotnou tvorbu embeddingu a jeho výslednou velikost. Nabízelo se hned několik možných přístupů, jak výstupní embedding vytvořit.

Pro první test jsem po posledním CMT bloku (viz obrázek 3.6) aplikoval pouze konvoluční vrstvu s proměnným počtem konvolučních filtrů. Výsledkem této vrstvy byly mapy příznaků o rozměrech 6×6. Tyto mapy jsem narovnal do jednoho embeddingu, který reprezentoval vstupní obrázek.

Při druhém testu jsem zachoval konvoluční vrstvu z prvního testu, na jejíž výsledek jsem aplikoval batch normalizaci a 2D average pooling. To znamená, že pokud měl výstup konvoluční vrstvy rozměry $C \times H \times W$, pak výstup pooling vrstvy měl rozměr C.

Třetím a posledním testem bylo přidání plně propojené vrstvy za 2D average pooling. Tato vrstva obsahuje další trénovatelné parametry a bylo tedy možné, že výstupem bude lepší reprezentace vstupního obrázku.

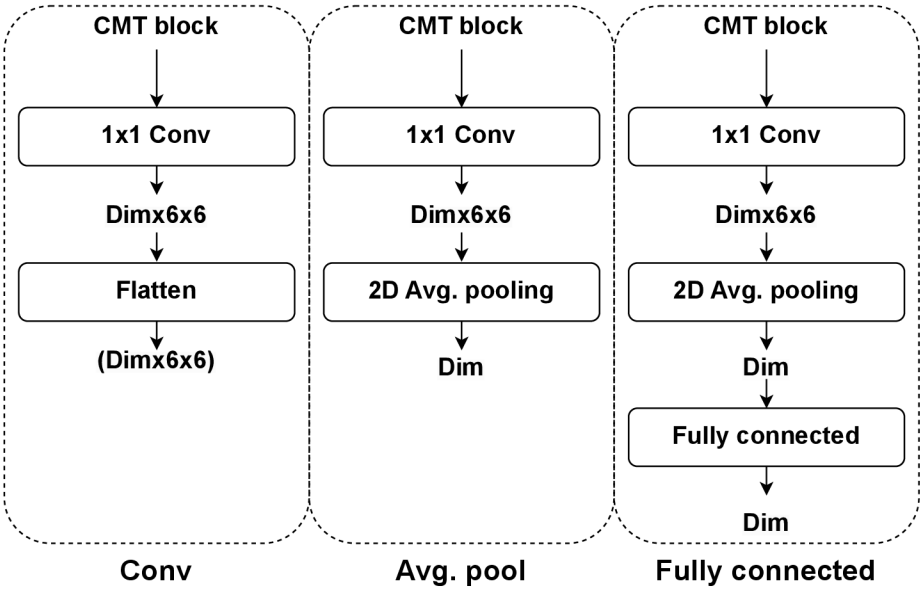
Trénování probíhalo se stejným nastavením jako předchozí experimenty. Tabulka 5.15 shrnuje dosažené výsledky. Obrázek 5.7 znázorňuje vyzkoušené postupy tvorby embeddingu.

Embeddingy tvořené pouze konvoluční vrstvou jsou v reálných aplikacích téměř nepoužitelné, a to z důvodu jejich velikosti. Např. embedding z konvoluční vrstvy s výstupem $512 \times 6 \times 6$ má konečný rozměr 18 432. Důvodem jejich otestování bylo zjištění, zda a jak moc dochází ke ztrátě informací reprezentující vstupní obrázek, při následné redukci rozměru. Z tabulky je patrné, že embeddingy s redukovaným rozměrem si zachovávají dostatečné informace o reprezentaci obrázku.

Nejlepších výsledků jsem dosáhl s použitím konvoluční vrstvy následující average poolingem. Na datasetu VeRi-776 mělo přidání plně propojené vrstvy za následek jen malé zhoršení, ovšem na datasetu VehicleID plně propojená vrstva výsledky zhoršila znatelně. Na datasetu VeRi-776 byla optimální velikost výstupního embeddingu 1 280. Na datasetu

Dataset	Embedding	mAP	Rank1	Rank5
VeRi-776	$256 \times 6 \times 6$ - konv.	0,681	85,5	94,3
VeRi-776	$512 \times 6 \times 6$ - konv.	0,722	93,3	97,1
VeRi-776	$768 \times 6 \times 6$ - konv.	0,725	94,2	97,2
VeRi-776	256 - pooling	0,739	94,2	97,3
VeRi-776	512 - pooling	0,730	94,0	97,3
VeRi-776	768 - pooling	0,742	**94,8**	97,8
VeRi-776	1 280 - pooling	**75,7**	94,7	**98,2**
VeRi-776	256 - plně propojená	0,697	93,1	97,1
VeRi-776	512 - plně propojená	0,730	92,4	97,0
VeRi-776	768 - plně propojená	0,732	93,4	96,7
VehicleID-2400	$256 \times 6 \times 6$ - konv.	—	62,6	86,0
VehicleID-2400	$512 \times 6 \times 6$ - konv.	—	66,2	88,4
VehicleID-2400	$768 \times 6 \times 6$ - konv.	—	74,2	91,2
VehicleID-2400	256 - pooling	—	70,8	89,7
VehicleID-2400	512 - pooling	—	74,3	**92,1**
VehicleID-2400	768 - pooling	—	71,2	91,5
VehicleID-2400	1 280 - pooling	—	**74,4**	91,6
VehicleID-2400	256 - plně propojená	—	62,4	87,2
VehicleID-2400	512 - plně propojená	—	63,6	86,4
VehicleID-2400	768 - plně propojená	—	66,3	87,5

Tabulka 5.15: Výsledky modelu CMT-xs s různými způsoby tvorby výstupního embeddingu. Pro větší přehlednost tabulky uvádím u datasetu VehicleID pouze podmnožinu o 2 400 identitách.



Obrázek 5.7: Vyzkoušené možnosti tvorby embeddingu.

VehicleID jsem dosáhl s embeddingem o rozměru 1 280 nejlepšího výsledku na metrice

Rank1. Proto jsem se rozhodl nadále využívat konvoluční vrstvu následovanou 2D average poolignem a výsledným embeddingem o rozměru 1 280.

Postprocessing

Dosavadní experimenty neobsahovaly žadný postprocessing. Postprocessing dokáže znatelně zlepšit úspěšnost modelů při fázi testování. Pro datasety VeRi-776 a VehicleID jsem vyzkoušel dvě metody postprocessingu.

První metodou byl tzv. *augmentation test*. Z každého testovacího obrázku jsem extrahoval dva embeddingy. První embedding byl z neupraveného obrázku a druhý z horizontálně otočeného obrázku. Pro výsledný embedding jsem oba embeddingy zprůměroval.

Druhou metodou byl reranking. Výsledky na datasetech VeRi-776 a VehicleID s postprocessingem jsou v tabulce 5.16. Na dataset VehicleID nebyl použit reranking, protože pro query vstup je v gallery vždy jen jedna shoda.

Dataset	Reranking	Aug. test	mAP	Rank1	Rank5
VeRi-776	—	—	0,757	94,7	**98,2**
VeRi-776	—	✓	0,763	94,8	97,2
VeRi-776	✓	—	**0,860**	**95,6**	97,5
VeRi-776	✓	✓	**0,860**	**95,6**	97,3
VehicleID-800	—	—	—	82,7	97,0
VehicleID-800	—	✓	—	**83,0**	**97,6**
VehicleID-1600	—	—	—	76,4	94,2
VehicleID-1600	—	✓	—	**77,7**	**94,9**
VehicleID-2400	—	—	—	**74,4**	91,6
VehicleID-2400	—	✓	—	74,2	**92,1**

Tabulka 5.16: Výsledky modelu CMT-xs na datasetech VeRi-776 a VehicleID s využitím postprocessing metod.

Na datasetu VeRi-776 výsledky na metrice mAP nejvíce zvýšil reranking. Při použití pouze augmentation testu se také dosáhlo zlepšení, ale v kombinaci s rerankingem byly výsledky stejné jako při použití samotného rerankingu. Augmentation test na datasetu VehicleID výsledky zlepšil jen lehce.

Experimenty na CityFlow a CarsReId74k

Dosavadní experimenty probíhaly na datasetech VehicleID a VeRi-776. Bylo to z důvodu jejich velikosti, kdy nebyl časový problém provést více trénování. Datasety CarsReId74k a CityFlow v kombinaci s VehicleX jsou mnohem větší datasety, a proto jsem se rozhodl na nich spustit testování v momentě, kdy budu mít důkladně model CMT otestovaný na prvních dvou zmíněných datasetech.

Trénování modelu CMT-xs na datasetu CityFlow společně s VehicleX proběhlo v 80 epochách s AdamW optimalizátorem. Velikost batch byla nastavena na 128. Vývoj loss hodnoty při trénování znázorňuje obrázek 5.8 (a). Po natrénování jsem adaptoval UDA dotrénování, stejně jako při modelu TransReID. Během inference se provedl reranking a augmentation test. Dosažené výsledky jsou v tabulce 5.17.

Trénování na CarsReId74k probíhalo v 30 epochách, jinak bylo totožné s předchozím trénováním. Vývoj loss hodnoty je znázorněn na obrázku 5.8. Výsledky jsou v tabulce 5.18.

Stejně jako u modelu TransReID jsem při evaluaci datasetu CityFlow využil přístup image-to-track. Postprocessing se zde ukázal jako zásadní. S jeho použitím se výsledek zlepšil o 0,188 na metrice mAP, kde překonal i mnou natrénovaný model TransReID o 0,005.

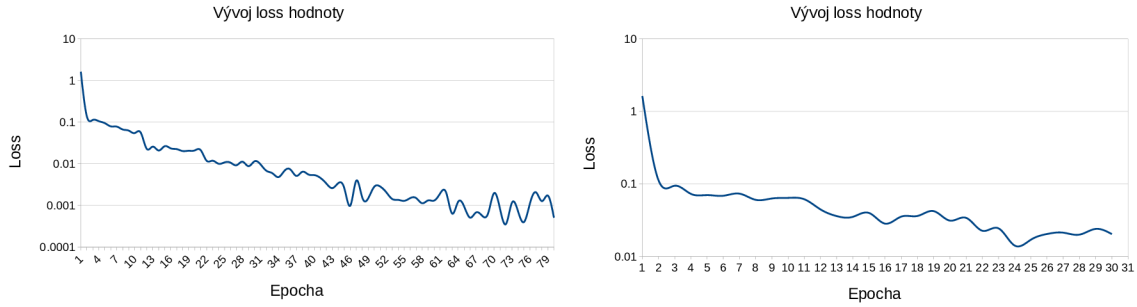
I při evaluaci datasetu CarsReId74k jsem dodržel doporučení využít image-to-track přístup. Model CMT-xs dopadl na všech metrikách lépe než model TransReID.

Dataset	Reranking	UDA	Aug. test	mAP	Rank1	Rank5
CityFlow	—	—	—	0,240	52,2	**62,7**
CityFlow	✓	—	—	0,382	49,7	50,1
CityFlow	✓	✓	—	0,390	50,3	50,4
CityFlow	✓	✓	✓	**0,428**	**52,3**	54,5

Tabulka 5.17: Výsledky modelu CMT-xs na datasetu CityFlow.

Dataset	mAP	Rank1	Rank5
CarsReId74k	0,699	59,3	83,2

Tabulka 5.18: Výsledky modelu CMT-xs na datasetu CarsReId74k.



(a) CityFlow + VehicleX

(b) CarsReId74k

Obrázek 5.8: Vývoj loss hodnoty při trénování na kombinaci datasetů CityFlow a VehicleX (a). Vývoj loss hodnoty při trénování na datasetu CarsReId74k (b).

5.4 Porovnání se state-of-the-art modely

Tabulka 5.19 nabízí porovnání mezi mými natrénovanými modely a state-of-the-art modely pro re-identifikaci vozidel. Model TransReID s využitím klíčových bodů a postprocessingu se přiblížil k úspěšnosti state-of-the-art modelům a některé i překonal. Model CMT-xs s využitím postprocessingu dosáhl nejlepšího výsledku na datasetu VeRi-776 na metrice mAP, čímž prokázal, že je vhodný pro úlohu re-identifikaci vozidel. Stejně tak na datasetu VehicleID dosáhl nejlepšího výsledku na metrice Rank5. Na datasetu CityFlow byly výsledky obou modelů srovnatelné. V porovnání s výsledky účastníků soutěže AI City

Challenge ovšem znatelně zaostávaly. Je ale potřeba mít na paměti, že např. vítěz soutěže využil dohromady celkem 16 natrénovaných modelů. Na datasetu CarsReId74k byl model CMT-xs úspěšnější než model TransReID. Rozdíl na metrice mAP činil 0,048. V porovnání s modelem LFTD - WE [24], který jako jediný byl také testován na tomto datasetu, byl výsledek modelu CMT-xs na metrice mAP o 0,08 horší.

Model	VeRi-776		VehicleID-800		CityFlow	CarsReId74k	
	mAP	Rank1	Rank1	Rank5	mAP	mAP	Rank1
SAN [21]	0,725	93,3	79,7	94,3	—	—	—
VANet [3]	0,663	89,8	**88,1**	97,3	—	—	—
PGAN [35]	0,793	96,5	77,5	92,1	—	—	—
SAVER [11]	0,796	96,4	79,9	95,2	—	—	—
TANet [13]	0,805	95,4	82,9	95,7	—	—	—
HPGN [23]	0,802	96,7	83,9	—	—	—	—
TransReID (orig.) [7]	0,805	**96,8**	85,2	97,5	0,565	—	—
DMT [17]	—	—	—	—	**0,744**	—	—
NewGeneration [9]	—	—	—	—	0,715	—	—
Cybercore [8]	—	—	—	—	0,613	—	—
LFTD - WE [24]	—	—	—	—	—	**0,779**	**71,3**
TransReID	0,767	96,3	77,6	94,4	0,423	0,651	54,7
TransReID (*kb*)	0,637	89,4	—	—	—	—	—
TransReID (*kb*, *post*)	0,748	91,1	—	—	—	—	—
CMT-xs	0,757	94,7	82,7	97,0	0,240	0,699	59,3
CMT-xs (*post*)	**0,860**	95,6	83,0	**97,6**	0,428	—	—

Tabulka 5.19: Porovnání dosažených výsledků se state-of-the-art modely. Pro dataset CityFlow jsou místo názvů modelů názvy týmů ze soutěže AI City Challenge 2021. Označení *kb* znamená, že model využíval klíčových bodů. Označení *post* znamená, že byl využit postprocessing.

Kapitola 6

Závěr

Cílem mojí diplomové práce bylo detailní nastudování a experimentování s moderními přístupy k re-identifikaci vozidel založených na vision transformerech. K experimentování jsem si vybral dva modely, které jsem testoval na čtyřech různých datových sadách (VeRi-776, VehicleID, CityFlow a CarsReId74k).

Prvním modelem, kterým jsem se zabýval, byl model TransReID. Tento model je založen čistě na vision transformerech a byl vytvořen pro účely re-identifikace vozidel. Nejprve jsem jej natrénoval na dostupných datasetech a porovnal s výsledky původních autorů. Další a hlavní část experimentů s tímto modelem byla věnována klíčovým bodům, jejichž anotace jsou dostupné pro dataset VeRi-776. Cílem bylo zjištění, zda se vision transformer dokáže natrénovat i s využitím jen oblastí okolo nejdůležitějších bodů vozidla. Pro každý klíčový bod byla extrahována oblast o rozměru 16×16 okolo tohoto bodu a z těchto oblastí byl sestaven vstupní obrázek. Při tomto přístupu se model může zaměřit jen na nejvýznamnější části vozidla a také se ze vstupního obrázku odstraní šum, který by jinak mohl být okolo vozidla. Druhým efektem tohoto přístupu je snížení výpočetní náročnosti, jelikož model pracuje s méně vstupy. Výsledky s využitím klíčových bodů se oproti neupravenému modelu propadly o téměř 25% na metrice mAP. Tento propad nastal ze dvou důvodů. Prvním důvodem byl fakt, že ne všechny klíčové body jsou na vozidle viditelné. Vstupní obraz je tak z velké části sestaven z neinformativních bodů. Tento problém jsem se pokusil vyřešit pomocí zrcadlení párových klíčových bodů. Druhým důvodem propadu byl fakt, že oblast o rozměru 16×16 byla příliš malá pro zachycení dostatku informací. Z toho důvodu jsem zvýšil velikost oblasti na 32×32. Při tomto řešení se propad snížil na 14,5% na metrice mAP. Po přidání postprocessingu výsledky na metrice mAP téměř dosáhly stejné úspěšnosti jako u původního modelu bez přidaného postprocessingu. Celkově se model TransReID s využitím klíčových bodů spolu s postprocessingem přiblížil k úspěšnosti state-of-the-art modelům a některé i překonal.

Druhým modelem byl model CMT. Tento model nebyl primárně vytvořen pro re-identifikaci vozidel. Mým cílem bylo ho upravit tak, aby se mohl rovnat se state-of-the-art modely. Nejefektivnější se projevilo trénování s triplet loss. Pro tvoření výstupního embeddingu jsem využil konvoluční vrstvu následovanou 2D average poolingem. Ideální velikost výstupního embeddingu byla 1 280. Model jsem otestoval na uvedených datasetech. Model si vedl velmi dobře na datasetu VeRi-776 v porovnání s ostatními state-of-the-art modely, kde na metrice mAP dosáhl nejlepšího výsledku. Na datasetu VehicleID měl nejlepší výsledek na metrice Rank5. Na datasetu CityFlow model CMT dosáhl podobných výsledků jako model TransReID. Model CMT překonal model TransReID na každé metrice na datasetu CarsReId74k.

Pro budoucí vývoj by bylo zajímavé vytvořit anotace klíčových bodů pro další datasety a více tak prozkoumat možnosti těchto bodů s modelem TransReID. Pro model CMT bych se zaměřil na využití dodatečných informací, jako např. ID kamery.

Literatura

[1] Carion, N., Massa, F., Synnaeve, G., Usunier, N., Kirillov, A. et al. *End-to-End Object Detection with Transformers*. arXiv, 2020. DOI: 10.48550/ARXIV.2005.12872. Dostupné z: https://arxiv.org/abs/2005.12872.

[2] Chen, H., Wang, Y., Guo, T., Xu, C., Deng, Y. et al. *Pre-Trained Image Processing Transformer*. arXiv, 2020. DOI: 10.48550/ARXIV.2012.00364. Dostupné z: https://arxiv.org/abs/2012.00364.

[3] Chu, R., Sun, Y., Li, Y., Liu, Z., Zhang, C. et al. *Vehicle Re-identification with Viewpoint-aware Metric Learning*. arXiv, 2019. DOI: 10.48550/ARXIV.1910.04104. Dostupné z: https://arxiv.org/abs/1910.04104.

[4] Dosovitskiy, A., Beyer, L., Kolesnikov, A., Weissenborn, D., Zhai, X. et al. *An Image is Worth 16x16 Words: Transformers for Image Recognition at Scale*. arXiv, 2020. DOI: 10.48550/ARXIV.2010.11929. Dostupné z: https://arxiv.org/abs/2010.11929.

[5] Guo, J., Han, K., Wu, H., Xu, C., Tang, Y. et al. CMT: Convolutional Neural Networks Meet Vision Transformers. *CoRR*. 2021, abs/2107.06263. Dostupné z: https://arxiv.org/abs/2107.06263.

[6] He, K., Zhang, X., Ren, S. a Sun, J. Deep Residual Learning for Image Recognition. *CoRR*. 2015, abs/1512.03385. Dostupné z: http://arxiv.org/abs/1512.03385.

[7] He, S., Luo, H., Wang, P., Wang, F., Li, H. et al. *TransReID: Transformer-based Object Re-Identification*. arXiv, 2021. DOI: 10.48550/ARXIV.2102.04378. Dostupné z: https://arxiv.org/abs/2102.04378.

[8] Huynh, S. V., Nguyen, N. H., Nguyen, N. T., Nguyen, V. T., Huynh, C. et al. A Strong Baseline for Vehicle Re-Identification. *CoRR*. 2021, abs/2104.10850. Dostupné z: https://arxiv.org/abs/2104.10850.

[9] Jiang, M., Zhang, X., Yu, Y., Bai, Z., Zheng, Z. et al. Robust Vehicle Re-Identification via Rigid Structure Prior. In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR) Workshops*. June 2021, s. 4026–4033.

[10] Khorramshahi, P., Kumar, A., Peri, N., Rambhatla, S. S., Chen, J.-C. et al. *A Dual-Path Model With Adaptive Attention For Vehicle Re-Identification*. arXiv, 2019. DOI: 10.48550/ARXIV.1905.03397. Dostupné z: https://arxiv.org/abs/1905.03397.

[11] Khorramshahi, P., Peri, N., Chen, J.-c. a Chellappa, R. *The Devil is in the Details: Self-Supervised Attention for Vehicle Re-Identification.* arXiv, 2020. DOI: 10.48550/ARXIV.2004.06271. Dostupné z: https://arxiv.org/abs/2004.06271.

[12] Kumar, R., Weill, E., Aghdasi, F. a Sriram, P. Vehicle Re-Identification: an Efficient Baseline Using Triplet Embedding. *CoRR.* 2019, abs/1901.01015. Dostupné z: http://arxiv.org/abs/1901.01015.

[13] Lian, J., Wang, D., Zhu, S., Wu, Y. a Li, C. Transformer-Based Attention Network for Vehicle Re-Identification. *Electronics.* 2022, sv. 11, č. 7. DOI: 10.3390/electronics11071016. ISSN 2079-9292. Dostupné z: https://www.mdpi.com/2079-9292/11/7/1016.

[14] Liu, H., Feng, J., Qi, M., Jiang, J. a Yan, S. End-to-End Comparative Attention Networks for Person Re-identification. *CoRR.* 2016, abs/1606.04404. Dostupné z: http://arxiv.org/abs/1606.04404.

[15] Liu, H., Tian, Y., Wang, Y., Pang, L. a Huang, T. Deep Relative Distance Learning: Tell the Difference Between Similar Vehicles. In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition.* 2016, s. 2167–2175.

[16] Liu, X., Liu, W., Mei, T. a Ma, H. PROVID: Progressive and Multimodal Vehicle Reidentification for Large-Scale Urban Surveillance. *IEEE Transactions on Multimedia.* 2018, sv. 20, č. 3, s. 645–658. DOI: 10.1109/TMM.2017.2751966.

[17] Luo, H., Chen, W., Xu, X., Gu, J., Zhang, Y. et al. *An Empirical Study of Vehicle Re-Identification on the AI City Challenge.* arXiv, 2021. DOI: 10.48550/ARXIV.2105.09701. Dostupné z: https://arxiv.org/abs/2105.09701.

[18] Luo, H., Gu, Y., Liao, X., Lai, S. a Jiang, W. Bag of Tricks and A Strong Baseline for Deep Person Re-identification. *CoRR.* 2019, abs/1903.07071. Dostupné z: http://arxiv.org/abs/1903.07071.

[19] Luo, H., Jiang, W., Zhang, X., Fan, X., Qian, J. et al. AlignedReID++: Dynamically matching local information for person re-identification. *Pattern Recognition.* 2019, sv. 94, s. 53–61. DOI: https://doi.org/10.1016/j.patcog.2019.05.028. ISSN 0031-3203. Dostupné z: https://www.sciencedirect.com/science/article/pii/S0031320319302031.

[20] Naphade, M., Wang, S., Anastasiu, D. C., Tang, Z., Chang, M.-C. et al. *The 5th AI City Challenge.* arXiv, 2021. DOI: 10.48550/ARXIV.2104.12233. Dostupné z: https://arxiv.org/abs/2104.12233.

[21] Qian, J., Jiang, W., Luo, H. a Yu, H. Stripe-based and Attribute-aware Network: A Two-Branch Deep Model for Vehicle Re-identification. *CoRR.* 2019, abs/1910.05549. Dostupné z: http://arxiv.org/abs/1910.05549.

[22] Schroff, F., Kalenichenko, D. a Philbin, J. FaceNet: A Unified Embedding for Face Recognition and Clustering. *CoRR.* 2015, abs/1503.03832. Dostupné z: http://arxiv.org/abs/1503.03832.

[23] Shen, F., Zhu, J., Zhu, X., Xie, Y. a Huang, J. Exploring Spatial Significance via Hybrid Pyramidal Graph Network for Vehicle Re-identification. *CoRR*. 2020, abs/2005.14684. Dostupné z: https://arxiv.org/abs/2005.14684.

[24] Spanhel, J., Sochor, J., Juránek, R., Dobes, P., Bartl, V. et al. Learning Feature Aggregation in Temporal Domain for Re-Identification. *CoRR*. 2019, abs/1903.05244. Dostupné z: http://arxiv.org/abs/1903.05244.

[25] Sun, Y., Zheng, L., Yang, Y., Tian, Q. a Wang, S. *Beyond Part Models: Person Retrieval with Refined Part Pooling (and a Strong Convolutional Baseline)*. arXiv, 2017. DOI: 10.48550/ARXIV.1711.09349. Dostupné z: https://arxiv.org/abs/1711.09349.

[26] Tang, Z., Naphade, M., Birchfield, S., Tremblay, J., Hodge, W. et al. *PAMTRI: Pose-Aware Multi-Task Learning for Vehicle Re-Identification Using Highly Randomized Synthetic Data*. arXiv, 2020. DOI: 10.48550/ARXIV.2005.00673. Dostupné z: https://arxiv.org/abs/2005.00673.

[27] Tang, Z., Naphade, M., Liu, M.-Y., Yang, X., Birchfield, S. et al. CityFlow: A City-Scale Benchmark for Multi-Target Multi-Camera Vehicle Tracking and Re-Identification. In: *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. June 2019, s. 8797–8806.

[28] Touvron, H., Cord, M., Douze, M., Massa, F., Sablayrolles, A. et al. *Training data-efficient image transformers & distillation through attention*. arXiv, 2020. DOI: 10.48550/ARXIV.2012.12877. Dostupné z: https://arxiv.org/abs/2012.12877.

[29] Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L. et al. *Attention Is All You Need*. arXiv, 2017. DOI: 10.48550/ARXIV.1706.03762. Dostupné z: https://arxiv.org/abs/1706.03762.

[30] Wang, G., Yuan, Y., Chen, X., Li, J. a Zhou, X. Learning Discriminative Features with Multiple Granularities for Person Re-Identification. In: *Proceedings of the 26th ACM international conference on Multimedia*. ACM, Oct 2018. DOI: 10.1145/3240508.3240552. Dostupné z: https://dl.acm.org/doi/10.1145/3240508.3240552.

[31] Wang, Z., Tang, L., Liu, X., Yao, Z., Yi, S. et al. Orientation invariant feature embedding and spatial temporal regularization for vehicle re-identification. In: *Proceedings of the IEEE international conference on computer vision*. 2017, s. 379–387. Dostupné z: https://openaccess.thecvf.com/content_ICCV_2017/papers/Wang_Orientation_Invariant_Feature_ICCV_2017_paper.pdf.

[32] Xuan, H., Stylianou, A., Liu, X. a Pless, R. Hard negative examples are hard, but useful. *CoRR*. 2020, abs/2007.12749. Dostupné z: https://arxiv.org/abs/2007.12749.

[33] Yao, Y., Zheng, L., Yang, X., Naphade, M. a Gedeon, T. *Simulating Content Consistent Vehicle Datasets with Attribute Descent*. arXiv, 2019. DOI: 10.48550/ARXIV.1912.08855. Dostupné z: https://arxiv.org/abs/1912.08855.

[34] Zakria, Deng, J., Khokhar, M. S., Aftab, M. U., Cai, J. et al. *Trends in Vehicle Re-identification Past, Present, and Future: A Comprehensive Review*. arXiv, 2021. DOI: 10.48550/ARXIV.2102.09744. Dostupné z: https://arxiv.org/abs/2102.09744.

[35] Zhang, X., Zhang, R., Cao, J., Gong, D., You, M. et al. *Part-Guided Attention Learning for Vehicle Instance Retrieval*. arXiv, 2019. DOI: 10.48550/ARXIV.1909.06023. Dostupné z: https://arxiv.org/abs/1909.06023.

[36] Zheng, J., Ramasinghe, S. a Lucey, S. Rethinking Positional Encoding. *CoRR*. 2021, abs/2107.02561. Dostupné z: https://arxiv.org/abs/2107.02561.

[37] Zheng, L., Shen, L., Tian, L., Wang, S., Wang, J. et al. Scalable Person Re-identification: A Benchmark. In:. Prosinec 2015, s. 1116–1124. DOI: 10.1109/ICCV.2015.133.

[38] Zheng, L., Yang, Y. a Hauptmann, A. G. Person Re-identification: Past, Present and Future. *CoRR*. 2016, abs/1610.02984. Dostupné z: http://arxiv.org/abs/1610.02984.

[39] Zheng, S., Lu, J., Zhao, H., Zhu, X., Luo, Z. et al. Rethinking Semantic Segmentation from a Sequence-to-Sequence Perspective with Transformers. In: *2021 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. 2021, s. 6877–6886. DOI: 10.1109/CVPR46437.2021.00681.

[40] Zheng, Z., Zheng, L. a Yang, Y. A Discriminatively Learned CNN Embedding for Person Re-identification. *CoRR*. 2016, abs/1611.05666. Dostupné z: http://arxiv.org/abs/1611.05666.

[41] Zhong, Z., Zheng, L., Cao, D. a Li, S. Re-ranking Person Re-identification with k-reciprocal Encoding. *CoRR*. 2017, abs/1701.08398. Dostupné z: http://arxiv.org/abs/1701.08398.

[42] Zhuang, Z., Wei, L., Xie, L., Zhang, H., Zhang, T. et al. Rethinking the Distribution Gap of Person Re-identification with Camera-based Batch Normalization. *CoRR*. 2020, abs/2001.08680. Dostupné z: https://arxiv.org/abs/2001.08680.