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

POČÍTÁNÍ VOZIDEL VE STATICKÉM OBRAZE

VEHICLE COUNTING IN STILL IMAGE

BAKALÁŘSKÁ PRÁCE
BACHELOR'S THESIS

AUTOR PRÁCE — ZDENĚK JELÍNEK
AUTHOR

VEDOUCÍ PRÁCE — Ing. JAKUB ŠPAŇHEL
SUPERVISOR

BRNO 2020

Ústav počítačové grafiky a multimédií (UPGM) Akademický rok 2019/2020

Zadání bakalářské práce

23070

Student: **Jelínek Zdeněk**
Program: Informační technologie
Název: **Počítání vozidel ve statickém obraze**
Vehicle Counting in Still Image
Kategorie: Zpracování obrazu
Zadání:

1. Seznamte se možnostmi počítání vozidel ve statickém obraze.
2. Vyhledejte literaturu zabývající se využitím konvolučních neuronových sítí pro počítání vozidel v obraze a zaměřte se na princip *"Počítání pomocí odhadu hustoty".*
3. Vyberte metody vhodné pro danou problematiku, implementujte s pomocí dostupných toolkitů a experimentujte s nimi.
4. Posbírejte vhodnou datovou sadu pro vyhodnocení vybraných metod.
5. Vhodným způsobem vyhodnoťte vybrané metody a diskutujte dosažené výsledky.
6. Vytvořte plakát a video prezentující vaší práci, její cíle a výsledky.

Literatura:

- Liu et al. "Context-Aware Crowd Counting." *CVPR,* 2019.
- Onoro-Rubio et al. "Towards perspective-free object counting with deep learning." *ECCV,* 2016.
- Dle pokynů vedoucího.

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

Vedoucí práce: **Špaňhel Jakub, Ing.**
Vedoucí ústavu: Černocký Jan, doc. Dr. Ing.
Datum zadání: 1. listopadu 2019
Datum odevzdání: 28. května 2020
Datum schválení: 1. listopadu 2019

Abstrakt

Hlavním cílem této práce bylo porovnání různých moderních přístupů k počítání vozidel pomocí odhadu hustoty. Celkem byly porovnány čtyři modely konvolučních neuronových sítí — Counting CNN, Hydra CNN, Perspective–Aware CNN a Multi–column CNN. Vyhodnocení natrénovaných modelů bylo provedeno na třech různých datasetech. Nejpřesnějších výsledků na všech datasetech dosáhl model Perspective–Aware CNN. Na datasetu PUCPR+ dosáhl hodnoty Mean Absolute Error 2,86, čímž prokázal, že jeho použití u problému počítání vozidel je vhodné.

Abstract

The main goal of this thesis was to compare different approaches to vehicle counting by density estimation. Four convolutional neural networks were tested — Counting CNN, Hydra CNN, Perspective–Aware CNN and Multi–column CNN. The evaluation of these models was done on three different datasets. The Perspective–aware CNN has achieved the most accurate results across all datasets. This model has reached 2.86 Mean Absolute Error on the PUCPR+ dataset, proving that it is the most suitable for the vehicle counting problem.

Klíčová slova

konvoluční neuronové sítě, mapa hustoty, odhad hustoty, počítání vozidel, Counting CNN, Hydra CNN, Perspective–Aware CNN, Multi–column CNN

Keywords

convolutional neural networks, density map, density estimation, vehicle counting, Counting CNN, Hydra CNN, Perspective–Aware CNN, Multi–column CNN

Citace

JELÍNEK, Zdeněk. *Počítání vozidel ve statickém obraze.* Brno, 2020. Bakalářská práce. Vysoké učení technické v Brně, Fakulta informačních technologií. Vedoucí práce Ing. Jakub Špaňhel

Počítání vozidel ve statickém obraze

Prohlášení

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

. . . . . .
Zdeněk Jelínek
25. května 2020

Poděkování

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

Obsah

1 Úvod **2**

2 Teorie neuronových sítí **3**
2.1 Trénování 5
2.2 Konvoluční neuronové sítě 6

3 Moderní přístupy k počítání vozidel v obraze **9**
3.1 Counting CNN 11
3.2 Hydra CNN 12
3.3 Perspective–Aware CNN 13
3.4 Multi–column CNN 15

4 Experimenty a trénování sítí **17**
4.1 Datasety 17
4.2 Generování ground truth map 19
4.3 Counting CNN 21
4.4 Hydra CNN 22
4.5 Perspective–Aware CNN 22
4.6 Multi–column CNN 23

5 Vyhodnocení experimentů **25**
5.1 Výsledky modelů na datasetu TRANCOS 26
5.2 Výsledky modelů na datasetu CARPK 28
5.3 Výsledky modelů na datasetu PUCPR+ 29
5.4 Porovnání rychlostí 30
5.5 Porovnání map hustot 31

6 Závěr **34**

Literatura **35**

Kapitola 1

Úvod

Ve své bakalářské práci se zabývám jedním z mnoha problémů počítačového vidění, a sice počítáním vozidel v obraze. Během let vznikla řada metod jak řešit počítání objektů, respektive vozidel. Nejmodernější přístupy jsou založeny na konvolučních neuronových sítích, které dosahují state–of–the–art výsledků. Zaměřil jsem se především na takové sítě, které k problému počítání vozidel přistupují jako k úloze odhadu hustoty. Výstupem takovýchto sítí je mapa hustoty, která poskytuje informace nejen o počtu vozidel, ale i jejich rozmístění v obraze.

Pro úspěšnou implementaci konvolučních neuronových sítí je potřeba znát jejich principy a teorií. Tomuto tématu se věnuje kapitola 2.

V kapitole 3 probírám principy metod počítání objektů a jak se vyvíjely postupem času. Další částí této kapitoly je popis vybraných konvolučních neuronových sítí, se kterými jsem experimentoval. Celkem jsem vybral čtyři sítě (Counting CNN, Hydra CNN, Perspective–Aware CNN a Multi–column CNN), každá s jinou architekturou a jiným přístupem k problému počítání objektů v obraze.

Následující kapitola 4 obsahuje detailní popis použitých datových sad a proces trénování a testování jednotlivých sítí.

Kapitola 5 se věnuje popisu vyhodnocovacích metrik, vyhodnocení provedených experimentů a porovnání úspěšností natrénovaných sítí. V této kapitole také nabízím porovnání rychlostí modelů a příklady vygenerovaných map hustot vozidel.

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

Kapitola 2

Teorie neuronových sítí

Neuronové sítě jsou výpočetním modelem, schopným aproximace jakékoli spojité funkce f [1]. Neuronová síť je tvořena tzv. umělými neurony. Tyto neurony jsou základní výpočetní jednotkou sítí. Model neuronu je uveden na obrázku 2.1.

Neuron má libovolný počet vstupů x_i a jeden výstup y. Každý vstup má přiřazenou váhu w_i. Hodnoty vstupů se násobí s odpovídajícími váhami. Tímto způsobem mohou být vstupní hodnoty posilovány nebo naopak potlačovány. Váha tedy představuje důležitost daného vstupu. Po vynásobení se všechny hodnoty sečtou. K tomuto součtu je pak ještě přičtena prahová hodnota (*bias*). Získaný výsledek je transformován aktivační funkcí, která zajišťuje nelinearitu. Matematický zápis výpočtu výstupu neuronu je definován vztahem 2.1.

$$y = f(\sum_{i=1}^{N} w_i x_i + w_0) \tag{2.1}$$

Neurony jsou uspořádány do vrstev. Rozlišují se tři typy vrstev: vstupní vrstva, skrytá vrstva a výstupní vrstva. Vstupní vrstva přijímá všechna externí vstupní data a předává je dalším vrstvám. Vstupy skryté vrstvy jsou výstupy vstupní vrstvy, nebo výstupy předchozí skryté vrstvy. Jestliže má neuronová síť vícero skrytých vrstev, mluvíme o *hluboké neuronové síti* (*deep neural network*). Poslední výstupní vrstva predikuje výsledek. Obrázek 2.1 znázorňuje příklad neuronové sítě se dvěma skrytými vrstvami. Sítě, jejichž vrstvy netvoří žádné cyklické spoje, se nazývají dopředné [4].

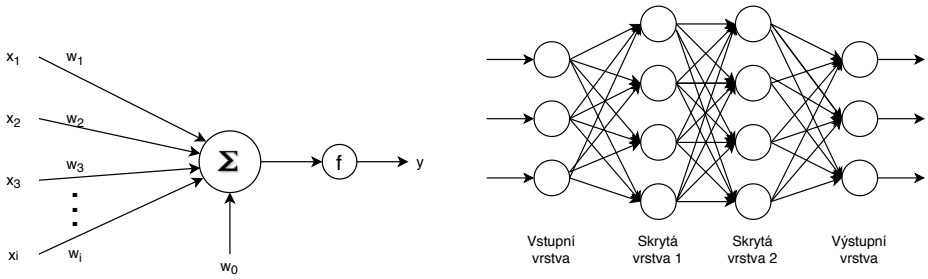
Aktivační funkce

Aktivační funkce řídí výstup neuronů a především do nich zavádí nelinearitu. Výběr správné aktivační funkce má velký vliv na chování a rychlost učení dané sítě. Funkce rozhoduje, zda by měl být neuron aktivní nebo ne. V dalších sekcích popíšu několik nejpoužívanějších aktivačních funkcí.

Sigmoida

Sigmoida je nelineární aktivační funkce, někdy též zvaná jako *logistická* funkce. Nejčastěji je využívaná v dopředných neuronových sítích. Sigmoida je definována vztahem 2.2. Její definiční obor je na množině reálných čísel a obor hodnot je na intervalu $(0, 1)$. Nejčastějším využitím sigmoidy je určení pravděpodobnosti daného jevu [11].

$$f(x) = \frac{1}{(1 + e^{-x})} \tag{2.2}$$



Obrázek 2.1: Model neuronu (levý obrázek) a příklad dopředné neuronové sítě se dvěma skrytými vrstvami (pravý obrázek).

Hyperbolický tangens

Hyperbolický tangens, někdy také *tanh* funkce, je aktivační funkce velmi podobná sigmoidě. Jejím oborem hodnot je interval $(-1, 1)$. Hyperbolický tangens je definován vztahem 2.3.

Výhodou jejího použití oproti sigmoidě je fakt, že negativní vstupy budou namapovány na negativní hodnoty a nulové vstupy budou namapovány na hodnoty blízké nule [14]. Hyperbolický tangens se nejčastěji využívá v rekurentních neuronových sítích pro zpracování přirozeného jazyka a rozpoznání řeči [11].

$$f(x) = \left(\frac{e^x - e^{-x}}{e^x + e^{-x}}\right) \tag{2.3}$$

Softmax

Aktivační funkce *softmax* se nejvíce využívá u klasifikačních úloh. Softmax přijímá na vstupu vektor reálných čísel, které normalizuje tak, aby jejich součet byl roven jedné. Její výstup lze považovat jako rozdělení pravděpodobnosti nad n třídami. Softmax je matematicky vyjádřena vztahem 2.4 [14].

Nejčastěji se tato aktivační funkce objevuje ve výstupních vrstvách neuronové sítě. Hlavní rozdíl mezi sigmoidou a funkcí softmax je ten, že sigmoida se využívá v úlohách binární klasifikace, zatímco softmax v úlohách, kde se klasifikuje vícero tříd [11].

$$f(x_i) = \frac{exp(x_i)}{\sum_j exp(x_j)} \tag{2.4}$$

Rectified Linear Unit

Rectified Linear Unit (ReLU) je jednou z nejpoužívanějších aktivačních funkcí. Jedna z výhod ReLU je rychlost jejího výpočtu (neobsahuje dělení, nebo jiné výpočetně náročné operace). ReLU se využívá především v konvolučních neuronových sítích. Tato aktivační funkce je definována vztahem 2.5 [11].

$$f(x) = max(0, x) = \begin{cases} x & \text{pro } x \geq 0 \\ 0 & \text{pro } x < 0 \end{cases} \tag{2.5}$$

2.1 Trénování

Trénování (učení) neuronových sítí je iterativní proces, během kterého se mění hodnoty vah na jednotlivých spojeních mezi neurony. Principem trénování neuronových sítí je nalezení takových hodnot vah, aby se minimalizovala hodnota chybové funkce (viz 2.1) [4].

K trénování neuronových sítí lze přistupovat několika způsoby, z nichž nejznámější a nejpoužívanější jsou trénování s učitelem a bez učitele. Hlavní rozdíl mezi nimi spočívá v tom, že při trénování s učitelem se využívá tzv. *ground truth* dat, jinými slovy, předem známe výstupní hodnoty pro vstupní data. Za pomocí těchto ground truth dat se trénovaný model snaží naučit funkci, která nejlépe aproximuje vztah mezi vstupními daty a jejich odpovídajícími výstupy. Na druhou stranu, trénování bez učitele nevyužívá žadných předem známých výstupních dat. Principem této metody trénování je tzv. *shlukování* (*clustering*), při kterém se model snaží najít vztahy mezi daty sám [17].

Před samotným trénováním je nutné určit hodnoty tzv. *hyperparametrů*. Hyperparametry řídí chování učících se algoritmů. Jejich hodnoty se algoritmus nemůže naučit, a proto se musí zadat ručně. Po dobu trénování se jejich hodnoty nemění. Příklady hyperparametrů, jejichž hodnoty je třeba nastavit, jsou počet *epoch* (jedna epocha znamená jednu iteraci přes celý dataset), chybová funkce nebo hodnota dávky (*batch size*) [4].

V první fázi trénování je proveden dopředný průchod (*forwardpropagation*). Neuronovou sítí projdou trénovací data napříč všemi vrstvami, kde každý neuron provede svou transformaci, a v poslední vrstvě se vygenerují odhadované výstupy [4].

Za pomocí chybové funkce se porovnají vygenerované odhadované výstupy s očekávanými. Po spočítání chybové funkce se tato hodnota využije při algoritmu zvaném zpětný průchod (*backpropagation*). Při zpětném průchodu se informace z chybové funkce propaguje od výstupní vrstvy až po vrstvu vstupní. Během zpětného průchodu se za pomocí hodnoty chybové funkce počítají gradienty, které se využívají při aktualizaci vah. *Gradient* je spočítán jako parciální derivace chybové funkce v závislosti na dané váze [4].

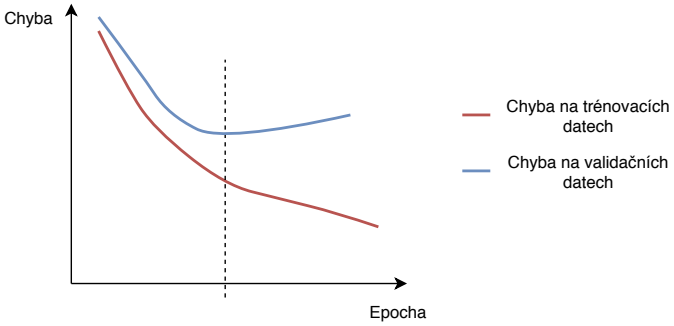
Pro aktualizaci vah se využívá optimalizačního algoritmu zvaného *gradientní sestup* (*gradient descent*). Pro danou váhu se použije opačná hodnota gradientu, protože gradient vždy udává směr pro zvýšení hodnoty chybové funkce. Aktualizace vah je matematicky vyjádřena vztahem 2.6.

$$w_{i+1} = w_i - \epsilon \nabla_{w_i} f(w_i), \tag{2.6}$$

kde w_i jsou hodnoty vah před aktualizací, w_{i+1} jsou aktualizované hodnoty vah, ϵ je parametr učení zvaný *learning rate*, ∇_{w_i} je gradient a f je chybová funkce. Algoritmus gradientního sestupu se opakuje, dokud hodnota chybové funkce nepřestane klesat, nebo dokud nepřesáhne požadovanou hodnotu. Jedna z možných variant algoritmu gradientního sestupu je tzv. *Stochastický gradientní sestup* (*Stochastic Gradient Descent*, SGD), který k aktualizaci vah používá hodnotu chybové funkce spočítanou pro každý trénovací vzorek zvlášť. Za zmínku stojí i optimalizační algoritmus Adam, jehož hodnota learning rate se v průběhu trénování mění [4].

Přeučení

Během trénování neuronových sítí může dojít k jevu zvanému přeučení (*overfitting*), kdy trénovaný model ztrácí schopnost generalizace, protože si zapamatovává vlastnosti a vztahy specifické pro trénovací data. K tomuto jevu dochází při příliš dlouhém trénování, nebo v případě, kdy je v trénovací sadě málo dat. Grafické znázornění přeučení je na obrázku 2.2.



Obrázek 2.2: Jev zvaný přeučení. Ačkoliv hodnota chyby na trénovací množině dat stále klesá, chyba na validační množině dat po několika epochách klesat přestane. V ten moment dochází k přeučení a model přestává správně generalizovat.

Jedna z možností, jak poznat, kdy u sítě dochází k přeučení, je rozdělit trénovací data na dvě disjunktní množiny — trénovací a validační. Data z trénovací množiny se použijí k trénování modelu a data z validační množiny se využijí k monitorování přeučení modelu. Po skončení každé epochy se spočítá hodnota chybové funkce na validačních datech, aniž by se měnily hodnoty vah. Pokud chyba na validačních datech přestane klesat, znamená to, že dochází k přetrénování.

Chybová funkce

Chybová funkce (*loss function*, nebo *cost function*) porovnává výstup neuronové sítě s očekávaným (ground truth) výsledkem. Její hodnota vyjadřuje, jak dobré nebo špatné jsou predikce neuronové sítě. Čím nižší hodnota chybové funkce, tím lepší jsou predikce.

Jedna ze základních chybových funkcí je *mean squared error* (MSE). MSE je definována vztahem 2.7.

$$MSE = \frac{1}{N}\sum_{i=1}^{N}(\hat{Y}_i - Y_i)^2, \tag{2.7}$$

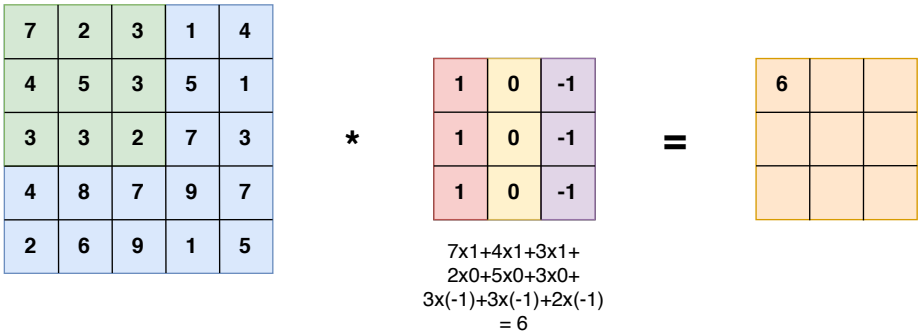
kde N je počet trénovacích dat, $\hat{Y}_i$ je ground truth výsledek a Y_i je odhadovaný výsledek [4].

2.2 Konvoluční neuronové sítě

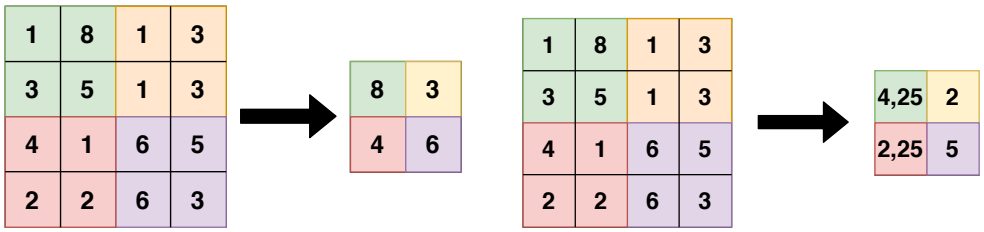
Konvoluční neuronové síě (CNN) jsou alternativním druhem neuronových sítí, které se nejčastěji využívají ke zpracování dat v podobě obrázků. Rozdílem mezi klasickými a konvolučními neuronovými sítěmi je ten, že v konvolučních sítích se využívá tzv. konvoluční vrstva. CNN typicky obsahují tři typy vrstev — *konvoluční* vrstvu, *aktivační* vrstvu a *pooling* vrstvu. V následujících sekcích tyto vrstvy popíšu.

Konvoluční vrstva

Konvoluční vrstva je jádrem konvolučních neuronových sítí. Základem konvoluční vrstvy je matematická operace konvoluce. Účelem konvoluce je získání příznaků (vlastností) z ob-



Obrázek 2.3: Grafické znázornění operace konvoluce nad 2D obrázkem.



Obrázek 2.4: Max–pooling (levý obrázek) a average–pooling (pravý obrázek).

rázku. Pro dvourozměrný obrázek a dvourozměrný *konvoluční filtr* (kernel) je operace konvoluce definována vztahem 2.8.

$$S(i,j) = (I * K)(i,j) = \sum_m \sum_n I(m,n)K(i-m, j-n), \tag{2.8}$$

kde I je vstupní obrázek, K je konvoluční filtr, i a j definují bod na obrázku a m a n jsou rozměry konvolučního filtru [4]. Grafické znázornění operace konvoluce je na obrázku 2.3.

Konvoluční vrstva obsahuje sadu konvolučních filtrů, typicky menších než vstupní obrázek. Během dopředného průchodu se každý filtr konvoluuje se vstupním obrázkem přes všechny jeho dimenze (délka, šířka a hloubka). Výstupem tohoto procesu je tzv. mapa příznaků (*feature map*). Pro každý filtr se vytvoří jedna mapa příznaků. Pro použití konvolučního filtru je nutné specifikovat jeho délku, šířku, hloubku a tzv. krok (*stride*), jenž určuje, jak se filtr bude pohybovat po obraze (např. při použití kroku o velikosti 1, se bude filtr pohybovat s posunem jednoho pixelu).

Pooling vrstva

Principem pooling vrstvy je sjednocení několika hodnot v mapě příznaků v dané oblasti do jedné, čímž dojde ke snížení jejích rozměrů a tím i ke snížení počtu parametrů sítě. Tato oblast je vytvořena pomocí filtru, který se po mapě příznaků pohybuje s určitým krokem (stride). Je–li velikost kroku větší než jedna, dojde ke snížení rozměrů mapy příznaků. Nejčastěji se volí filtr o velikosti 2×2 s krokem 2 [4]. S takovýmto filtrem by se mapa příznaků snížila o 75%.

Proces sjednocení hodnot do jedné má několik variant. Jednou z nich je tzv. *max–pooling*, který z dané oblasti vybere největší hodnotu a tu propaguje dál. Druhou je tzv.

average–pooling, jenž z hodnot v dané oblasti spočítá průměr a ten propaguje dále [4]. Princip max–poolingu a average–poolingu je znázorněn na obrázku 2.4

Jedna z výhod použití pooling vrstvy je ta, že zajišťuje invarianci proti malému posunutí vstupních dat [4].

Kapitola 3

Moderní přístupy k počítání vozidel v obraze

Jedním z odvětví počítačového vidění je počítání objektů v obraze. I přes to, že nejrozšířenějšími objekty počítání jsou lidé v davech, techniky, které byly pro tento problém vyvinuty lze aplikovat i na podobné problémy, jako například počítání vozidel [18].

Pro přesný odhad počtu a hustoty objektů je nutno překonat několik překážek jakými jsou například nerovnoměrné rozmístění objektů v obraze, různá perspektiva ve scéně nebo nerovnoměrné osvětlení. V posledních několika letech se techniky pro počítání objektů vyvinuly z dřívějších metod, které byly často limitovány na malé změny hustoty objektů, na techniky, jež dokážou úspěšně predikovat počet i hustotu v různých scénách s různým počtem objektů [18].

Výzkumníci se nejdříve snažili řešit počítání a odhad hustoty objektů použitím různých metod, jako například pomocí detekce, shlukování nebo regrese [8]. Počáteční přístupy k metodám počítání pomocí regrese využívaly převážně ručně tvořené *příznaky* (*features*). S nástupem konvolučních neuronových sítí se tyto ručně tvořené příznaky ukázaly jako nedostatečné. Konvoluční neuronové sítě zaznamenaly natolik výrazné zlepšení, že v dnešní době se většina práce zaměřuje právě na ně [18].

Jak jsem již naznačil v úvodu, mým zaměřením je počítání pomocí odhadu hustoty objeků. Rozdíl oproti běžnému počítání objektů je ten, že odhad hustoty se snaží namapovat vstupní obrázek na jeho odpovídající mapu hustoty, která nabízí detailnější informace o rozložení objektů ve scéně. Nyní uvedu několik možných přístupů k odhadu hustoty.

Lempitsky a spol. [7] navrhli učící se algoritmus lineárního mapování mezi sadou příznaků z určité oblasti obrázku a odpovídající mapou hustoty, čímž zakomponovali prostorovou informaci do procesu učení. Tímto novým přístupem se vyhnuli obtížnému úkolu naučení se detekce a lokalizace jednotlivých instancí objektu.

Pham a spol. [13] si všimli, že samotné lineární mapování je těžké na naučení. Proto navrhli metodu nelineárního mapování mezi sadou příznaků z určité oblasti obrázku a mapou hustoty. Využili k tomu regresi za pomocí *náhodného lesu* (*random forest*). Jedním z předmětů jejich zkoumání byl problém velkého rozdílu mezi vzhledem a tvarem v oblastech, ve kterých je hustota objektů vysoká a v oblastech, ve kterých je hustota objektů naopak nízká. Kvůli tomuto rozdílu by trénování na datové sadě obsahující snímky hustě zaplněné scény i nezaplněné scény mělo být těžší, než trénování několika různých lesů pro každou scénu. Proto zavedli pojem přeplněnost (crowdedness), který vyjadřuje vlastnost zaplnění

scény ve snímku. Natrénovali dva náhodné lesy v závislosti na přeplněnosti, s nimiž dosáhli state–of–the–art výsledků.

V reakci na úspěchy konvolučních neuronových sítí v řadě odvětví počítačového vidění, se výzkumníci pokusili využít jejich schopností naučit se nelinearní mapování obrázků na jejich odpovídající mapy hustoty nebo odpovídající počet objektů. Během let vznikla řada přístupů založených na CNN, které lze kategorizovat podle vlastností sítě na:

Základní CNN: Modely zahrnující základní konvoluční vrstvy. Tyto modely byly jedny z prvních, jež využívali hlubokého učení pro pročítání a odhad hustoty objektů. Příkladem základní CNN pro počítání lidí v davu je [3].

Scale–aware CNN: Sofistikovanější modely sítí, které jsou odolné vůči rozdílům v měřítku. Tato odolnost je dosáhnuta různými technikami, např. použitím vícero sloupců v síti. Příkladem scale–aware CNN jsou [21, 12, 2].

Context–aware CNN: Přístup, který se snaží zakomponovat lokální a globální kontextové informace v obrázku pro dosažení menších chyb v odhadech. Příkladem context–aware CNN je [16].

Multi-task CNN: Modely, které se snaží zkombinovat počítání vozidel společně s dalšími úlohami (např. odhadem rychlosti davu). Příkladem takovéto sítě je [22].

Jedna z dalších možností kategorizace je podle způsobu inference:

Inference na vzorcích z obrázku: Při použití tohoto přístupu se konvoluční neuronové sítě trénují na vzorcích (patches) z obrázku. Testování probíhá za využití posuvného okénka (sliding window), které extrahuje vzorky z obrázku. Pro každý vzorek se získá odpovídající predikce. Pro získání konečného počtu objektů se predikce agregují. Příkladem takovéto sítě je [12].

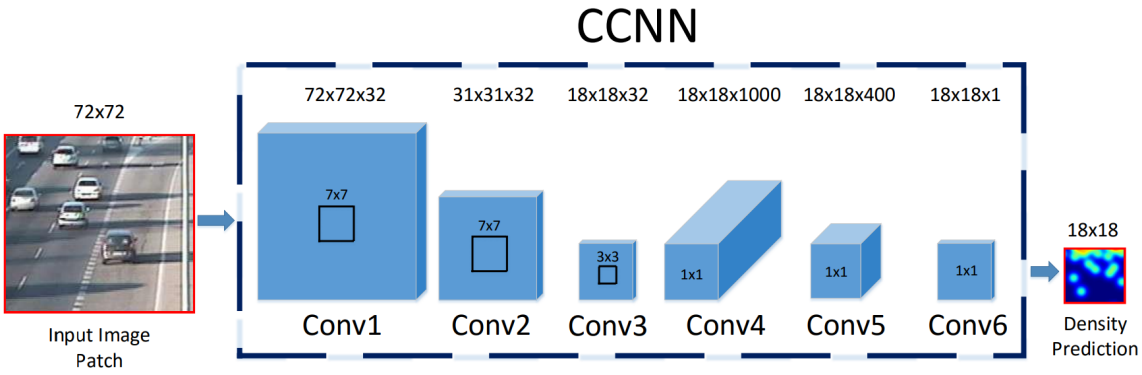
Inference na celém obrázku Tento přístup se vyhýbá použití výpočetně drahého posuvného okénka. Inference probíhá na celém obrázku bez jakéhokoli extrahování vzorků. Příkladem takovéto sítě je [10].

Wang a spol. [19] byli jedni z prvních, kdo aplikoval konvoluční neuronové sítě k úloze odhadu hustoty. Wang a spol. navrhli konvoluční neuronovou síť pro počítání lidí v extrémně hustém davu. V architektuře navrženého modelu využili část sítě AlexNet [6], kde poslední plně propojenou vrstvu o 4096 neuronech nahradili jedním neuronem, který odhadoval počet lidí.

Zhang a spol. [20] analyzovali existující metody a zjistili, že výsledky těchto metod se drasticky sníží, použijí–li se na novou scénu, která se neobjevila v trénovacím datasetu. Aby překonali tento problém, natrénovali svoji síť tak, aby řešila dvě spolu související úlohy: odhad počtu a odhad hustoty. Navrhovaný model byl nejprve předtrénován na všech trénovacích scénách. Aby síť dokázala dobře predikovat i z dosud neviděných scén, je model dotrénován na trénovacích datech, které se podobají cílové scéně. Síť byla vyhodnocena na datech s různými scénami i s jednou scénou. V obou případech dosáhla state–of–the–art výsledků.

Během let byla vytvořena řada přístupů jak odhadovat počet a hustotu v obraze. Obecně přístupy využívající konvoluční neuronové sítě překonávají tradiční přístupy napříč datasety a jsou prokazatelně efektivní u hustě zaplněných scén s různými podmínkami [18].

V dalších sekcích popisuji přístupy, které jsem se rozhodl reimplementovat a otestovat.



Obrázek 3.1: Architektura sítě Counting CNN. Převzato z [12].

3.1 Counting CNN

Tento model vychází z práce *Towards perspective–free object counting with deep learning* [12]. *Counting CNN* (CCNN) je plně konvoluční neuronová síť schopná přesné regrese map hustot objektů ze vzorků obrázku (image patches). Jedna z velkých výhod tohoto modelu je ta, že nepotřebuje k regresi map hustoty žádné mapy perspektivy nebo jiné geometrické informace o scéně. Další výhodou CCNN je skutečnost, že k natrénování nepotřebuje dvou, či více chybových funkcí, nebo dodatečného regresoru. Přímým výstupem CCNN je odhadovaná mapa hustoty. Tento model byl vytvořen pro účely počítání lidí i vozidel.

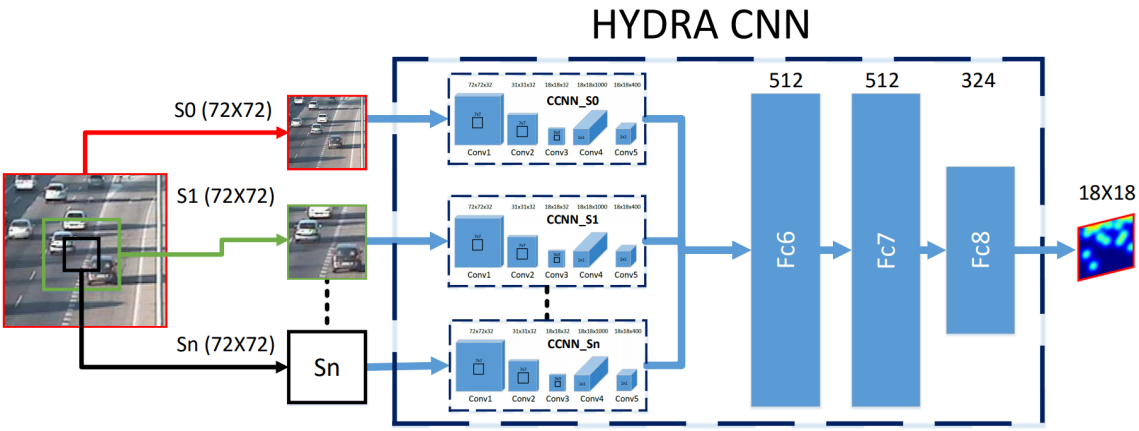
Architekturu modelu lze vidět na obrázku 3.1. Síť je složena z šesti konvolučních vrstev. Vstupem je část obrázku o rozměrech 72×72. První i druhá konvoluční vrstva mají filtry o velikosti 7×7 s hloubkou 32. Obě jsou následovány max–pooling vrstvami s velikostí filtru 2×2. Třetí konvoluční vrstva má filtr o velikosti 5×5 a hloubkou 64 a je také následována max–pooling vrstvou s filtrem o velikosti 2×2. Čtvrtá i pátá vrstva mají obě filtr s velikostí 1×1 o hloubce 1000, respektive 400. Tento model neobsahuje žádnou plně propojenou vrstvu, čímž vznikla plně konvoluční neuronová síť. Každá konvoluční vrstva je následována aktivační funkcí ReLU. Poslední šestá vrstva má filtr o velikosti 1×1 s hloubkou 1. Tato vrstva má na starost vytvoření predikce mapy hustoty $D_{pred}^{(P)}$ pro vstupní část obrázku P.

K natrénování tohoto modelu využili autoři chybovou funkci zvanou *Euclidean regression loss*, která je definována vztahem 3.1.

$$l(\Omega) = \frac{1}{2N} \sum_{n=1}^{N} \left\| \mathcal{R}(P_n|\Omega) - D_{gt}^{P_n} \right\|_2^2, \tag{3.1}$$

kde N odpovídá počtu vzorků obrázků v trénovací dávce (training batch), $D_{gt}^{P_n}$ představuje ground truth mapu hustoty pro konkrétní část obrázku P_n. Ω označuje parametry sítě. Autoři využili k natrénování optimalizační algoritmus *Stochastic Gradient Descent* (SGD).

Samotný odhad mapy hustoty probíhá následovně. Z testovacího obrázku se prvně extrahují části (patches), které slouží jako vstup do CCNN. Jak uvádí obrázek 3.1, dimenze vstupu je pevně daná na velikost 72×72, proto se nejdřív musí extrahované části upravit na tuto velikost. Jelikož CCNN model obsahuje dvě max–pooling vrstvy, je výstupní mapa hustoty 1/4 velikosti vstupního obrázku, tedy 18×18. Při zvětšení odhadované mapy hustoty ovšem dochází ke změně počtu vozidel v mapě. Proto musí být nově vytvořená mapa hustoty normalizována. Normalizace je definována vztahem 3.2.



Obrázek 3.2: Architektura sítě Hydra CNN. Převzato z [12].

$$\hat{D}_{pred} = \frac{\sum_{\forall p} D^{P}_{pred}(p)}{\sum_{\forall p} \hat{D}^{P}_{pred}(p)} \hat{D}_{pred} \tag{3.2}$$

Po normalizaci všech map hustot následuje poslední krok, jenž spočívá ve spojení map do jedné.

CCNN byla testována na úloze počítání vozidel. Porovnání mých výsledků a výsledků autorů CCNN nabízím v dalších kapitolách.

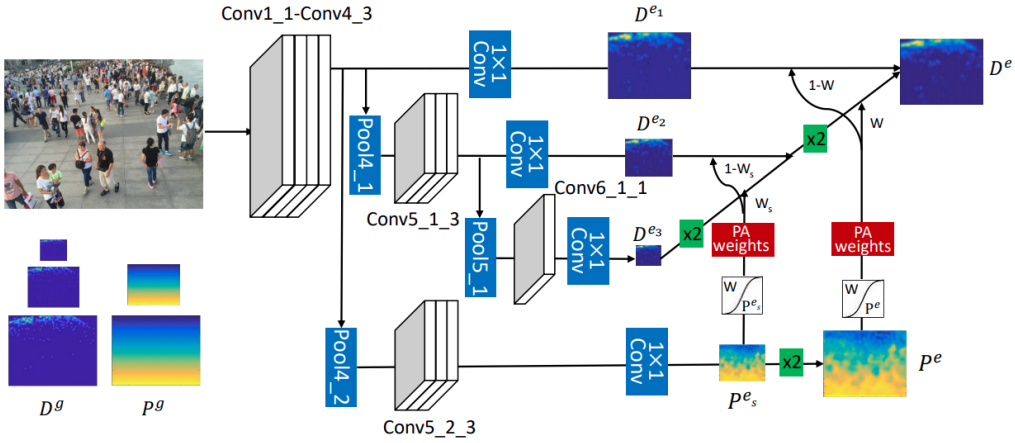
3.2 Hydra CNN

Tento model vychází z práce *Towards perspective–free object counting with deep learning* [12]. V klasickém modelu pro počítání za pomocí regrese, se většinou používá geometrická korekce vstupních příznaků, například za pomocí map perspektivy. Tato korekce by měla pomoci řešit problém extrahovaných příznaků ze stejného objektu v různých hloubkách scény. Takovéto příznaky bývají velmi odlišné ve svých hodnotách, důsledkem čehož model vykazuje horší výsledky. Bylo popsáno v několika pracích, že geometrická korekce výsledky zlepšuje [7, 8, 20].

Model *Hydra CNN* se snaží problém geometrické korekce řešit po svém. Model má tzv. *scale–aware* architekturu, která žádné korekce nevyužívá. Ke správnému odhadu hustoty využívá informace z vícero naškálovaných vstupů zároveň. Dle autorů by tento model měl být schopen přesně odhadnout počet objektů i v dosud neviděných scénách s různou perspektivou.

Architektura modelu Hydra CNN je rozdělena na dvě části. První částí jsou tzv. *hlavy* Hydry. Každá hlava je zodpovědná za naučení se reprezentace konkrétního naškálovaného vstupu s_i. Výstupem hlav jsou sady příznaků popisující různě naškálované vstupní obrázky. Všechny tyto příznaky jsou spojeny a předány dál druhé části Hydry. Tou je *tělo*. Tělo je tvořeno několika plně propojenými vrstvami a je zodpovědné za vygenerování výsledného odhadu mapy hustoty.

Celková architektura je znázorněna na obrázku 3.2. Z něj je možné si všimnout, že pro hlavy Hydry $(CCNN_s0, ..., CCNN_sn)$ byl použit lehce upravený model sítě Counting CNN 3.1, kterému byla odstraněna poslední konvoluční vrstva. Výstupy hlav jsou konkatenovány a předány tělu, které je tvořeno dvěma plně propojenými vrstvami. Obě mají



Obrázek 3.3: Architekura modelu PACNN. Převzato z [10].

512 neuronů a jsou následovány ReLU vrstvami. Poslední je plně propojená vrstva s 324 neurony, jejímž výstupem je odhadovaná mapa hustoty.

Stejně jako model CCNN, i Hydra byla natrénována za použití chybové funkce *Euclidean regression loss* a algoritmu *Stochastic Gradient Descent*.

Proces predikce mapy hustoty je podobný jako u CCNN. Nejdříve se z testovacího obrázku extrahují jeho části. Hydra může mít 1 až n vstupů. Počet vstupů odpovídá počtu hlav Hydry. Jestliže má Hydra dva vstupy (tedy dvě hlavy), pak se na první vstup přivede nijak nezměněná čast obrázku o rozměrech 72×72 a na druhý vstup se přivede o polovinu oříznutá část obrázku, která se zvětší na rozměry 72×72. Výstupní mapy hustoty jsou pak sjednoceny do jedné.

Hydra byla vyhodnocena na problému počítání vozidel. Autoři otestovali Hydru se dvěma, třemi a čtyřmi hlavami. Nejlepších výsledků dosáhla Hydra se třemi hlavami. Porovnání jejich výsledků s mými nabízím v dalších kapitolách.

3.3 Perspective–Aware CNN

Tento model vychází z práce *Revisiting Perspective Information for Efficient Crowd Counting* [10]. *Perspective–Aware CNN* (PACNN) je konvoluční neuronová síť, vytvořená pro efektivní počítání lidí v davu za využití odhadu hustoty. Tento model do svého odhadu zahrnuje i informaci o perspektivě scény v podobě mapy perspektivy, která by měla síti pomoci lokalizovat osoby na malých oblastech v obrázku. Ačkoliv je tento model určený pro počítání lidí v davu, pokusil jsem se ho aplikovat na problém počítání vozidel.

Architektura PACNN je znázorněna na obrázku 3.3. První čtyři konvoluční vrstvy (*Conv1_1* až *Conv4_3*) jsou převzaté z předtrénované sítě *VGG16*. *VGG16* je konvoluční neuronová síť, vytvořená ke klasifikaci obrázků, která dosáhla přesnosti 92.7% na datasetu ImageNet (přes 14 milionů obrázků s 1 000 třídami) [15]. Z výstupu vrstvy *Conv4_3* se model rozděluje na dvě části, které provádí regresi mapy hustoty, respektive mapy perspektivy.

PACNN vytváří tři odhady map hustoty z výstupů vrstev *Conv4_3*, *Conv5_1_3*, *Conv6_1_1*. Odhadované mapy hustoty jsou označeny D^{e_1}, D^{e_2} a D^{e_3}. Díky různě velikým konvolučním filtrům, by se jejich kombinací měla síť přizpůsobit různě velikým objektům v obrázku. Z důvodu použití max–pooling vrstev jsou mapy v menším rozlišení než vstupní

obrázek. Odhadovaná mapa hustoty D^{e_1} má velikost 1/8 vstupního obrázku, D^{e_2} má velikost 1/16 a D^{e_3} má velikost 1/32. Pro získání výsledného odhadu mapy hustoty se mapa D^{e_3} zvětší ($Up()$) na velikost mapy D^{e_2} a obě mapy se zprůměrují. Nově vytvořená mapa se zvětší na velikost D^{e_1} a znovu se obě zprůměrují, čímž vznikne výsledná mapa hustoty. Tento postup matematicky vyjadřuje vztah 3.3.

$$D^e = \frac{D^{e_1} + Up\left(\frac{D^{e_2}+Up(D^{e_3})}{2}\right)}{2} \tag{3.3}$$

Mapa perspektivy je vytvořena z výstupu vrstvy *Conv5_2_3* a je označena jako Pe_s. Pe_s má rozlišení 1/16 vstupního obrázku. Dále se tato mapa perspektivy zvětší na velikost 1/8 vstupního obrázku, čímž vznikne druhá mapa perspektivy P^e.

Jak jsem již zmínil, každá mapa hustoty byla vytvořena za použití jinak velikých konvolučních filtrů. Dle autorů by tedy mapa D^{e_1} měla dobře rozpoznávat malé hlavy, D^{e_2} středně velké hlavy a D^{e_3} velké hlavy. Aby se mapy perspektivy P^{e_s} a P^e mohly využít, přidaly se do sítě dvě tzv. *perspective–aware* (PA) váhovací vrstvy, které mají za úkol vytvořit z map perspektivy váhy. Tyto váhy se využijí při slučování map hustot. Myšlenka je taková, že pokud je daná mapa hustoty dobrá v detekci menších hlav, pak PA vrstva umístí větší váhy na oblast s menší hlavou. Kombinace map hustot D^{e_2} a D^{e_3} je pak následovná:

$$D^{e_s} = (1 - W^s) \odot D^{e_2} + W^s \odot Up(D^{e_3}), \tag{3.4}$$

kde $\odot$ označuje Hadamardův součin a D^e_s výslednou mapu hustoty. W^s je výstupem *perspective–aware* váhovací vrstvy, jenž je získán použitím nelineární transformace $w^s_j = f(p^{e_s}_j)$ na hodnoty perspektivní mapy $p^{e_s}_j$. Zvolenou funkcí $f()$ byla sigmoida. Vztah 3.5 vyjadřuje, jak se z mapy perspektivy získají váhy.

$$w^s_j = f(p^{e_s}_j) = \frac{1}{1 + exp(-\alpha^s * (p^{e_s}_j - \beta^s))}, \quad w^s_j \in (0, 1), \tag{3.5}$$

kde α a β jsou trénovatelné parametry.

Kombinací map hustot D^{e_2} a D^{e_3} je vytvořena nová mapa hustoty D^{e_s}, která se kombinuje s mapu hustoty D^{e_1}. D^{e_s} se nejprve zvětší na velikost 1/8 vstupního obrázku a poté se sloučí s D^{e_1} s využitím druhé PA váhovací vrstvy.

$$D^e = (1 - W^s) \odot D^{e_1} + W^s \odot Up(D^{e_s}) \tag{3.6}$$

V porovnání s metodou průměrování, uvedenou vztahem 3.3, kde nejsou využity žádné váhy při kombinování map hustot, dává navržená metoda *perspective–aware* váhování různé váhy na mapy D^{e_1}, D^{e_2} a D^{e_3} v různých místech obrázku, tak, aby konečný výsledek byl odolný proti perspektivnímu zkreslení.

PACNN řeší dvě regresní úlohy. Regresi mapy hustoty a mapy perspektivy. Pro specifickou úlohu O se typicky využívá chybová funkce *mean squared error* (MSE), která sečte euklidovskou vzdálenost jednotlivých pixelů mezi odhadovanou mapou a ground truth mapou. MSE ovšem nebere v úvahu lokální korelaci v mapě. Z toho důvodu se ke trénování PACNN využilo chybové funkce DSSIM, která dokáže měřit rozdíly v lokálních vzorech na odhadované a ground truth mapě. Chybová funcke DSSIM L^{DSSIM} je odvozena od *structural similarity* (SSIM), která byla představena v práci *Image quality assessment: from error visibility to structural similarity* [23]. Celková chybová funkce pro úlohu O je dána vzta-

hem 3.7.

$$
\begin{aligned}
L_O(\Theta) &= L^{MSE} + \lambda L^{DSSIM} = \\
&= \frac{1}{2N}\sum_{i=1}^{N}\|E(X_i;\Theta) - G_i\|_2^2 + \lambda\frac{1}{N}\sum_{i=1}^{N}(1 - \frac{1}{M}\sum_j SSIM_i(j))
\end{aligned} \tag{3.7}
$$

$$
SSIM_i = \frac{(2\mu_{E_i}\mu_{G_i} + C_1)}{\mu_{E_i}^2 + \mu_{G_i}^2 + C_1} \cdot \frac{2\sigma_{E_iG_i} + C_2)}{\sigma_{E_i}^2 + \sigma_{G_i}^2 + C_2}, \tag{3.8}
$$

kde Θ je sada trénovatelných parametrů sítě PACNN. X_i je vstupní obrázek, N je počet trénovacích obrázků a M je počet pixelů v mapách. λ je váha, která balancuje L^{MSE} a L^{DSSIM}. E a G označuje odhadovanou mapu, respektive ground truth mapu pro danou úlohu O. Střední hodnoty (μ_{E_i}, μ_{G_i}) a směrodatné odchylky $(\sigma_{E_i}, \sigma_{G_i}, \sigma_{E_iG_i})$ ve funkci $SSIM_i$ jsou vypočítány použitím Gaussova filtru s rozptylem 1 v oblasti 5×5 na každé pozici j.

Pro úlohu regrese mapy perspektivy P, je chybová funkce L_P získána nahrazením parametrů E a G z 3.7 za P^e, respektive P^g, zatímco pro úlohu regrese mapy hustoty je chybová funkce L_D získána substitucí parametrů E a G za D^e a D^g. Celková chybová funkce je vyjádřena rovnicí 3.9.

$$
L = L_P + LD + \kappa L_{P^s} + \lambda_1 L_{D^1} + \lambda_2 L_{D^2} + \lambda_3 L_{D^3}. \tag{3.9}
$$

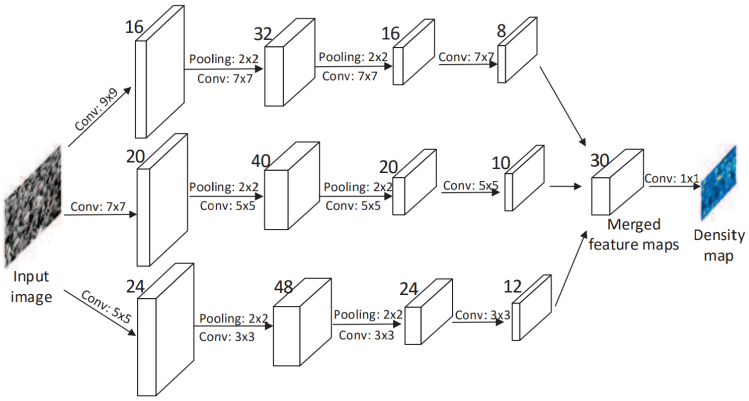
Pro testování není nutné vstupní obrázky dělit na části, architektura PACNN umožňuje vstupy o libovolných velikostech. PACNN byla vyhodnocena na problému počítání lidí, a z toho důvodu nelze porovnat mé výsledky s autorovými.

3.4 Multi–column CNN

Tento model vychází z práce *Single–Image Crowd Counting via Multi–Column Convolutional Neural Network* [21]. *Multi–column CNN* (MCNN) je konvoluční neuronová síť, která vznikla za účelem přesného odhadu počtu lidí v davu z obrázku s libovolnou hustotou osob a libovolnou perspektivou scény. Stejně jako PACNN (3.3) jsem se i MCNN pokusil aplikovat na problém počítání vozidel. Jednou z předních výhod architektury MCNN je, že vstupní obrázek může být jakékoli velikosti či rozlišení. MCNN je tvořena třemi konvolučnímy částmi — sloupci, které by měli kompenzovat absenci jakékoli informace o geometrii scény.

Obrázky s lidmi obvykle obsahují hlavy různých velikostí, a proto lze předpokládat, že konvoluční filtry se stejnými velikostmi by nebyly schopné se přizpůsobit charakteristikám davu v různých perspektivách. Z toho důvodu byla architektura MCNN rozdělena na tři paralelní sloupce, z nichž každý obsahuje konvoluční filtry s jinými velikostmi. Každý sloupec by se měl naučit rozpoznávat hlavy jiných velikostí. Například sloupec s většími filtry by měl být užitečnější pro rozpoznání větších hlav.

Celková architektura modelu MCNN je znázorněna na obrázku 3.4. Jak již bylo zmíněno, je tvořena třemi sloupci. Každý sloupec je strukturován stejným způsobem, liší se pouze v parametrech. Pro poolingové vrstvy byl zvolen max–pooling, každý s velikostí filtru 2×2. Za aktivační funkci byla zvolena ReLU, pro její dobré výsledky s konvolučními sítěmi. Výstupy všech tří sloupců jsou sjednoceny a pomocí konvoluční vrstvy s filtry 1×1 jsou přetvořeny na odhadovanou mapu hustoty. Kvůli použití max–pooling vrstev má výsledná mapa hustoty velikost 1/4 vstupního obrázku.



Obrázek 3.4: Architektura sítě MCNN. Převzato z [21].

K natrénovávání sítě zvolili autoři stejnou chybovou funkci a algoritmus jako autoři Counting CNN a Hydra CNN, tedy *Euclidean regression loss* a *Stochastic Gradient Descent*.

Jak jsem již uvedl, MCNN zvládá vstupy o libovolných velikostech a tím pádem se testování provádělo na neupravených testovacích obrázcích. MCNN bylo testováno pouze na datech obsahující osoby, a proto nebylo možné porovnat mé výsledky s autorovými.

Kapitola 4

Experimenty a trénování sítí

V této kapitole popisuji proces trénování a testování uvedených modelů. Každý model byl natrénován od základu, s výjimkou modelu PACNN (3.3), který využívá několik předtrénovaných vrstev ze sítě *VGG16*. Trénování probíhalo na datasetech, které popisuji v sekci 4.1.

Veškerá implementace byla provedena v jazyce Python 3[1]. K vytvoření konvolučních neuronových sítí jsem použil knihovnu Keras[2], která používá jako backend knihovnu TensorFlow[3], a která umožňuje akceleraci výpočtů na grafických kartách od společnosti NVIDIA. Pro manipulaci s obrázky jsem využil knihovnu scikit–image[4] a OpenCV[5]. U těch modelů, u kterých to bylo možné, jsem trénování akceleroval využitím grafické karty NVIDIA GeForce GTX 970[6] s pamětí o velikosti 4GB. V případech, kdy tato grafická karta nestačila, jsem trénování přesunul na *Google Colaboratory*[7]. Toto prostředí funguje ve webovém prohlížeči a samotné výpočty se provádí v cloudu. *Colaboratory* nabízí akceleraci pomocí GPU, ovšem samotný model GPU si nelze vybrat, je vám přidělen automaticky. Velkou nevýhodou *Colaboratory* je časový limit 12 hodin, při kterém je možno využívat přiřazenou kartu a provádět výpočty. Po skončení časového limitu je proces bez vyjímky ukončen. Tento limit lze zdvojnásobit zakoupením Pro verze Colaboratory. Této možnosti jsem využil. U možnosti Pro je také k dispozici až dvojnásobná paměť RAM.

Trénování modelů bylo ukončeno v momentě, kdy přestala pravidelně klesat hodnota validační chyby.

4.1 Datasety

V této sekci popíši datasety, které byly využity k natrénování a testování modelů. Celkem bylo využito tří datasetů — TRANCOS, CARPK a PUCPR+. Každá datová sada je složena ze snímků s různými scénami v různých podmínkách.

[1] https://www.python.org/
[2] https://keras.io/
[3] https://www.tensorflow.org/
[4] https://scikit-image.org/
[5] https://pypi.org/project/opencv-python/
[6] https://www.nvidia.com/cs-cz/geforce/900-series/
[7] https://colab.research.google.com/

Obrázek 4.1: Snímky z datasetu TRANCOS. Převzato z [5].

TRANCOS

TRaffic ANd COngestionS[8] (TRANCOS) je volně dostupná datová sada, která obsahuje snímky z různých dopravních situací pořízené sledovacími videokamerami. Dataset byl představen v práci *Extremely Overlapping Vehicle Counting* [5].

Dataset je tvořen 1 244 snímky, obsahující celkem 46 796 anotovaných vozidel. Vozidla byla anotována souřadnicemi středu vozidel. Dataset obsahuje snímky z různých scén se změnami v osvětlení a různou mírou překryvu vozidel. Snímky jsou v rozlišení 640×480 pixelů.

Sada je rozdělena na tři disjunktní sety. Trénovací, který obsahuje celkem 403 snímků, validační, obsahující 420 snímků a testovací se 421 snímky. Při trénování a vyhodnocování sítí jsem využil právě tohoto rozřazení.

Ke každému snímku (.jpg) je přiřazeno několik souborů. Jeden textový sobour (.txt), který definuje souřadnice pixelů, na kterých jsou středy vozidel. Dalším souborem je obrázek (.png) se stejným rozlišením jako snímek scény, obsahující zvýrazněné pixely definované v textovém souboru. Posledním je matlabový soubor (.mat) definující oblast zájmu (region of interest), která se má zohlednit při vyhodnocování sítí.

Příklady snímků z datasetu TRANCOS je možné vidět na obrázku 4.1.

CARPK

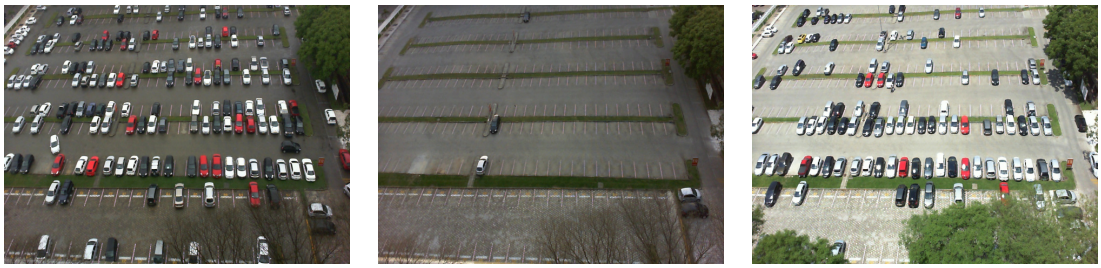
Car Parking Lot Dataset[9] (CARPK) je datová sada obsahující celkem 89 777 anotovaných vozidel ze čtyř různých parkovišť. Představena byla v práci *Drone–based Object Counting by Spatially Regularized Regional Proposal Networks* [9]. Snímky byly pořízeny dronem z výšky přibližně čtyřiceti metrů. Rozlišení všech snímků je 720×1280 pixelů.

Datová sada je rozdělena na trénovací set, který obsahuje 989 snímků s 42 274 anotovanými vozidly a testovací set, tvořen 459 snímky s 47 500 anotovanými vozidly. Vozidla jsou anotována pomocí tzv. bounding boxů. Bounding box je definován dvěma body. První bod označuje levý horní roh boxu a druhý bod označuje pravý dolní roh. Každému snímku je přiřazen soubor, který obsahuje souřadnice bounding boxů. CARPK nedefinuje validační set. Proto jsem trénovací set rozdělil na dva. Jeden sloužil k trénování sítí a obsahoval 890 snímků. Druhý sloužil k validaci a obsahoval 99 snímků.

Příklady snímků z datasetu CARPK je možné vidět na obrázku 4.2.

[8] Data jsou k dispozici na `http://agamenon.tsc.uah.es/Personales/rlopez/data/trancos/`
[9] Data jsou k dispozici na `https://lafi.github.io/LPN/`

Obrázek 4.2: Snímky z datasetu CARPK. Převzato z [9].



Obrázek 4.3: Snímky z datasetu PUCPR+. Převzato z [9].

PUCPR+

Pontifical Catholic University of Parana+ Dataset[10] (PUCPR+) je datová sada obsahující snímky z jednoho parkoviště o velikosti 331 míst. Tento dataset byl publikován společně s datasetem CARPK [9]. Snímky byly zachyceny z budovy v desátém patře, čímž byl vytvořen šikmý pohled na parkoviště. Snímky mají rozlišení 720×1280 pixelů.

Dataset je rozdělen na trénovací set o 100 snímcích a testovací set s 25 snímky. Celkový počet vozidel v trénovacím setu je 12 995 a v testovacím 3 920. Stejně jako CARPK, ani tento dataset nedefinuje rozdělení na trénovací data a validační. Rozdělil jsem tedy trénovací data na 75 trénovacích snímků a 25 validačních.

Příklady snímků z datasetu PUCPR+ je možné vidět na obrázku 4.3.

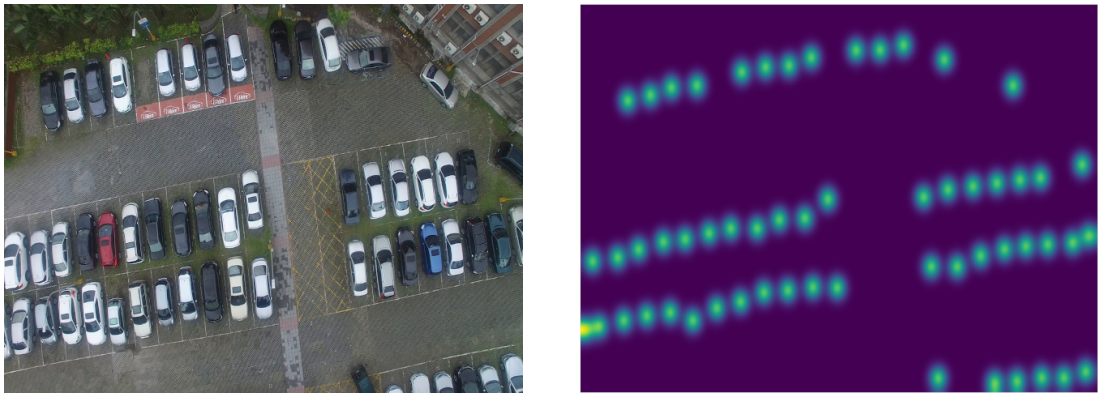
4.2 Generování ground truth map

K natrénování každého modelu je potřeba vytvořit k trénovacímu obrázku odpovídající ground truth mapu hustoty vozidel a u některých modelů i ground truth mapu perspektivy.

Mapa hustoty

Pro vytvoření ground truth mapy hustoty je potřeba mít trénovací obrázek vhodně anotovaný a to nejlépe souřadnicemi středu vozidla. Dataset TRANCOS využívá právě takovýchto anotací, kdežto datasety CARPK a PUCPR+ využívají anotací pomocí bounding boxů. To ovšem není problém, protože z bounding boxů lze lehce získat souřadnice středu boxu, které zároveň označují střed vozidla. Bounding box je dán dvěma souřadicemi, levým

[10] Data jsou k dispozici na https://lafi.github.io/LPN/



Obrázek 4.4: Snímek z datasetu CARPK a jeho odpovídající ground truth mapa hustoty.

horním rohem (x_i, y_i) a pravým dolním rohem (x_j, y_j). Střed bounding boxu (x, y) a tedy i vozidla je dán vztahem 4.1.

$$(x, y) = \left(\frac{x_i + x_j}{2}, \frac{y_i + y_j}{2} \right) \tag{4.1}$$

Ground truth mapu hustoty lze poté vytvořit za využití Gaussova filtru umístěného na získané souřadnice. Pro trénovací obrázek I je ground truth mapa hustoty definována vztahem 4.2,

$$D_I(p) = \sum_{\mu \in \mathbf{A}_I} \mathcal{N}(p; \mu, \Sigma) \tag{4.2}$$

kde $\mathbf{A}_I$ je sada 2D souřadnic anotovaných pro obrázek I. $\mathcal{N}(p; \mu, \Sigma)$ odpovídá výsledku aplikovaného Gaussova filtru, který generuje normální rozdělení, se střední hodnotou μ a kovarianční maticí Σ (zvolené hodnoty matice: $\Sigma = 15 \cdot 1_{2x2}$), umístěným na pixel, který byl definován bodem p. S takto definovanou mapou hustoty lze počet objektů získat sečtením všech hodnot mapy D_I, jak je uvedeno v rovnici 4.3. Obsah pod křivkou normálního rozdělení, které právě generuje Gaussův filtr, je vždy roven jedné, a protože hodnoty mapy sčítáme, je počet vozidel zachován i v případě, kdy se překrývají.

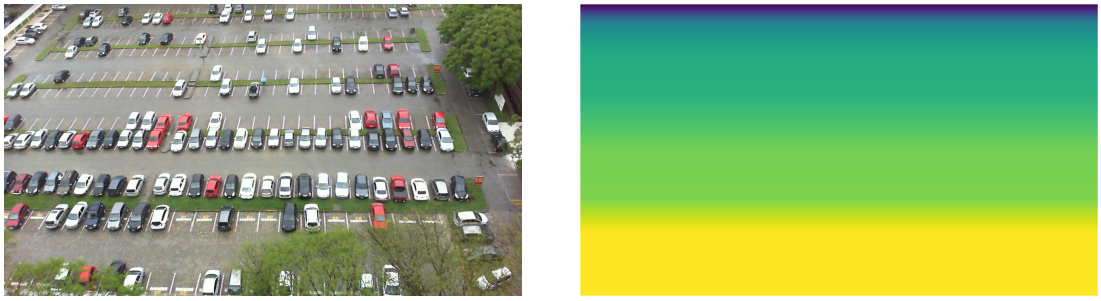
$$N_I = \sum_{p \in I} D_I(p) \tag{4.3}$$

Příkladem ground truth mapy hustoty je obrázek 4.4.

Mapa perspektivy

Mapu perspektivy lze vygenerovat pouze na datech, které byly anotovány pomocí bounding boxů. U datasetu TRANCOS tedy nebylo možné vytvořit grund truth mapy perspektivy. Dataset PUCPR+ nabízí pouze jeden pohled na scénu a tedy pro všechny snímky z této datové sady je mapa perspektivy stejná. Dataset CARPK sice obsahuje snímky s vícero scénami, jsou ale pořízeny se stejným pohledem na scénu (kolmo k zemi), a tedy i pro všechny snímky z této datové sady je mapa perspektivy stejná.

K vytvoření mapy perspektivy jsem použil podobný postup jako autoři práce *Cross–scene crowd counting via deep convolutional neural networks* [20]. Ti mapu perspektivy



Obrázek 4.5: Snímek z datasetu PUCPR+ a jeho odpovídající ground truth mapa perspektivy.

vytvořili tak, že vybrali několik různě umístěných osob (práce se týkala počítání osob) na obrázku a označili si jejich výšku na snímku. Hodnota perspektivy p_j^g na vybrané pozici j byla pak dána vztahem 4.4.

$$p_j^g = \frac{h_j}{H}, \tag{4.4}$$

kde h_j je označená výška člověka a H je průměrná výška člověka, jejíž hodnotu autoři určili jako $1,75$ metru. Tím získali několik hodnot perspektivy na daných řádcích obrázku. Hodnoty mapy perspektivy se mění pouze vertikálně. K vygenerování kompletní mapy perspektivy využili lineární funkci, jejímiž hodnotami vyplnili zbývající řádky mapy.

Můj postup se skládal z výběru vhodného snímku s dostatečným počtem vozidel různě rozmístěných ve scéně. Dalším krokem bylo manuální označení výšek několika vybraných vozidel. Za hodnotu H jsem vybral největší výšku vozidla ve scéně. Výslednou mapu perspektivy datasetu PUCPR+ lze vidět na obrázku 4.5.

4.3 Counting CNN

Ke trénování sítě Counting CNN jsem přistupoval podobně jako její autořix. Z každého trénovacího obrázku jsem náhodně extrahoval vzorky o velikostech 115×115. Z každého obrázku jsem vygeneroval 800 vzorků. Jelikož CCNN přijímá na vstupu obrázky o velikostech 72×72, musel jsem každý vzorek zmenšit na požadovanou velikost. Pro zvýšení počtu trénovacích dat jsem každý vzorek ještě horizontálně převrátil. Z jednoho obrázku jsem tedy vygeneroval celkově 1 600 vzorků. Stejný postup jsem aplikoval i u ground truth map hustot.

Parametry trénování byly nastaveny následovně. Za optimalizační algoritmus trénování byl zvolen *Stochastic Gradient Descent* s hodnotou learning rate nastavenou na hodnotu 0,001. Velikost dávky činila 128. Chybová funkce byla zvolena *Euclidean Regression loss*. Počet epoch nebyl stanoven napevno, jelikož samotné trénování trvalo několik dní. CCNN jsem natrénoval na každém z uvedených datasetů. CCNN trénovaná na datasetu TRANCOS dosáhla nejnižší hodnoty validační chyby po 98. epoše. Na datasetu CARPK bylo dosáhnuto nejnižší hodnoty po 112. epoše a na PUCPR+ po 94. epoše.

Při testování sítě se z každého testovacího obrázku extrahovaly vzorky obrázků metodou posuvného okénka s posuvem 10 pixelů. Ke každému vzorku CCNN vygenerovala odhad mapy hustoty. Celkový odhad hustoty se získal složením všech vygenerovaných částí dohro-

Obrázek 4.6: Příklad vstupních dat Hydry CNN.

mady. Ty části, které se překrývaly, a tedy se podílely na hodnotě hustoty v dané oblasti, jsem zprůměroval. Příklady výsledných odhadů map hustot lze vidět na obrázcích 5.5.

4.4 Hydra CNN

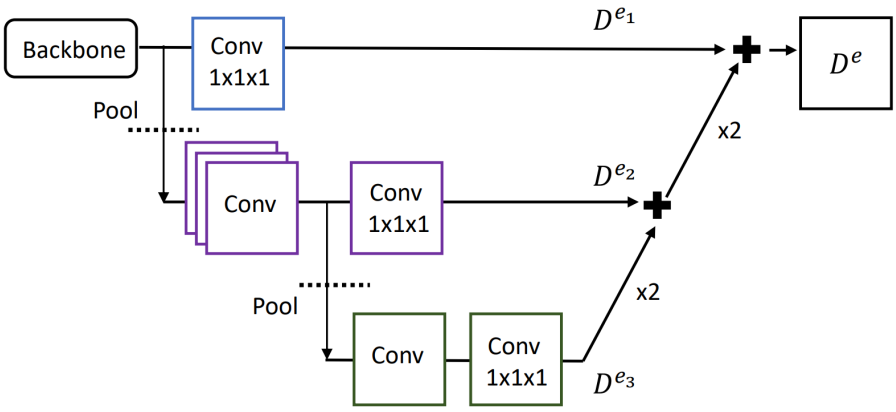
Hydra CNN z části využívá architekturu Counting CNN. Jak již bylo zmíněno v kapitole 3.2, Hydra je tvořena tzv. *hlavami* a *tělem*. Každá hlava, která funguje jako vstup sítě, je tvořena právě Counting CNN. Příprava dat k trénování je tedy velmi podobná, jako právě u samostatné CCNN. Nejlepších výsledů dosáhla původní Hydra se třemi hlavami. Z tohoto důvodu jsem ve všech natrénovaných verzích použil právě tento počet hlav. Z každého trénovacího obrázku jsem extrahoval 800 vzorků o velikostech 115×115, které jsem následně zmenšil na velikost 72×72 a taktéž horizontálně převrátil. Při použití tří vstupů (hlav) má vzorek, který jde do prvního vstupu, původní velikost extrahovaného vzorku, druhý je oříznut o 33% původní velikosti a třetí je oříznut o 66% původní velikosti. Oříznuté vzorky jsou poté zvětšeny na velikost 72×72. Příklad vstupních dat Hydry je uveden na obrázku 4.6. K natrénování tohoto modelu mi nestačila má hardwarová výbava, a proto jsem trénování přesunul na *Colaboratory*.

Hydra byla natrénována na každém datasetu zvlášť. Při trénování na datasetu TRANCOS byla dosáhnuta nejnižší validační chyba po 418. epoše, na datasetu CARPK po 493. epoše a na datasetu PUCPR+ po 444. epoše. Za chybovou funkci byla zvolena *mean squared error*. Velikost dávky byla 128. Za optimalizační algoritmus byl zvolen *Stochastic Gradient Descent* s hodnotou learning rate nastavenou na 0,001.

Při testování jsem z každého testovacího obrázku extrahoval jeho vzorky metodou posuvného okénka s posuvem 10 pixelů. Extrahované vzorky jsem rozdělil na tři části a naškáloval stejně jako při trénování. Proces složení vygenerovaných map hustot do jedné se shodoval s metodou uvedenou u Counting CNN.

4.5 Perspective–Aware CNN

Trénování sítě Perspective–aware CNN probíhalo ve dvou fázích. V první fázi se síť učila regresi mapy hustoty. To znamená, že z původní architektury sítě se odebrala část využívající perspektivní mapu. Architektura PACNN bez perspektivní mapy je zobrazena na obrázku 4.7. Výsledná mapa hustoty byla získána zprůměrováním třech výsledných odhadů map hustoty (podle vztahu 3.3). K natrénování bylo z každého trénovacího obrázku náhodně extrahováno 8 částí o velikostech 224×224, které posloužily jako vstup sítě. Tato velikost byla zvolena z důvodu využití sítě *VGG16*, která má právě tuto velikost vstupu. Ke každému



Obrázek 4.7: Architektura Perspective–aware CNN bez perspektivni části. Převzato z [10].

vstupnímu obrázku jsem vytvořil odpovídající ground truth mapu hustoty a z ní taktéž extrahoval 8 částí.

Parametry trénování byly nastaveny následovně. Velikost dávky byla 1. Trénování bylo optimalizováno za pomocí algoritmu *Stochastic Gradient Descent* s hodnotou learning rate nastavenou na 0,0001. V rámci experimentů jsem tuto část sítě natrénoval na všech datasetech uvedených v kapitole 4.1. Pro experimenty v druhé fázi trénování, jsem síť natrénoval i na datech zkombinovaných z datasetů CARPK a PUCPR+. Při trénování na datasetu TRANCOS dosáhla síť nejnižší hodnoty validační chyby po 803. epoše. Na datasetu CARPK byla dosažena nejnižší chyba po 684. epoše a na PUCPR+ po 520. epoše. Při kombinaci CARPK a PUCPR+ bylo dosaženo nejnižší chyby po 445. epoše. Za chybovou funkci byla zvolena DSSIM. V rámci experimentů jsem vyzkoušel natrénovat PACNN i s využitím chybové funkce *mean squeared error* na datasetu TRANCOS. Takto natrénovaná PACNN dávala ovšem horší výsledky než s chybovou funkcí DSSIM a proto jsem nadále využíval jen DSSIM.

Ve druhé fázi se regrese mapy hustoty optimalizovala pomocí mapy perspektivy. Do architektury sítě se tedy přidala část s perspektivní mapou. Optimalizace PACNN probíhala na modelech natrénovaných na datasetech PUCPR+ a na kombinaci datasetů CARPK a PUCPR+. Právě kombinace CARPK a PUCPR+ měla pomoci vyřešit nedostatek datasetu PUCPR+, jehož data nabízejí jen jednu variaci perspektivní mapy a tím zlepšit schopnost generalizace mapy perspektivy. Při dotrénování modelu, předtrénovaného z první fáze na datasetu PUCPR+, byla zaznamenána nejnižší hodnota chybové funkce po 620. epoše. Předtrénovaný model na kombinaci CARPK a PUCPR+ dosáhl nejnižší hodnoty 572. epoše. Ve fázi optimalizace pomocí perspektivní mapy jsem použil pouze chybovou funkci DSSIM, jelikož v porovnání s využitím *mean squared error* dosáhla PACNN s DSSIM lepších výsledků.

Architektura PACNN umožňuje testování obrázků libovolných velikostí bez nutnosti je rozdělovat na části a zpětně skládat, jak je tomu například u sítě Counting CNN. Testování obrázků se tím znatelně zrychlilo.

4.6 Multi–column CNN

Poslední implementovanou sítí byla Multi–column CNN. K natrénování bylo z každého trénovacího obrázku náhodně extrahováno 8 částí obrázku, které sloužily jako vstupní data sítě. Každá část obrázku měla velikost 224×224.

Stejně jako u předchozích modelů, i MCNN byla natrénována na každém datasetu zvlášť. Na datasetu TRANCOS dosáhla síť nejnižší hodnoty validační chyby po 537. epoše. Na CARPK po 412. epoše a na PUCPR+ po 576. epoše. K natrénování jsem využil chybovou funkci DSSIM. Velikost dávky byla 1. Za optimalizační algoritmus trénování byl zvolen *Stochastic Gradient Descent* s hodnotou learning rate nastavenou na 0,0001.

Stejně jako u PACNN, i u této sítě lze testovat snímky bez nutnosti je rozdělovat na vzorky a zpětně skládat.

Kapitola 5

Vyhodnocení experimentů

V této kapitole porovnávám natrénované modely, popsané v předchozí kapitole, a vyhodnocuji výsledky provedených experimentů. Obecně používanými vyhodnocovacími metrikami pro odhad hustoty jsou *Mean Absolute Error* (MAE) a *Mean Squared Error* (MSE). MAE a MSE jsou definovány vztahem 5.1 [10, 21, 20].

$$MAE = \frac{1}{N}\sum_{1}^{N}|z_i - \hat{z}_i|, \quad MSE = \sqrt{\frac{1}{N}\sum_{1}^{N}(z_i - \hat{z}_i)^2}, \tag{5.1}$$

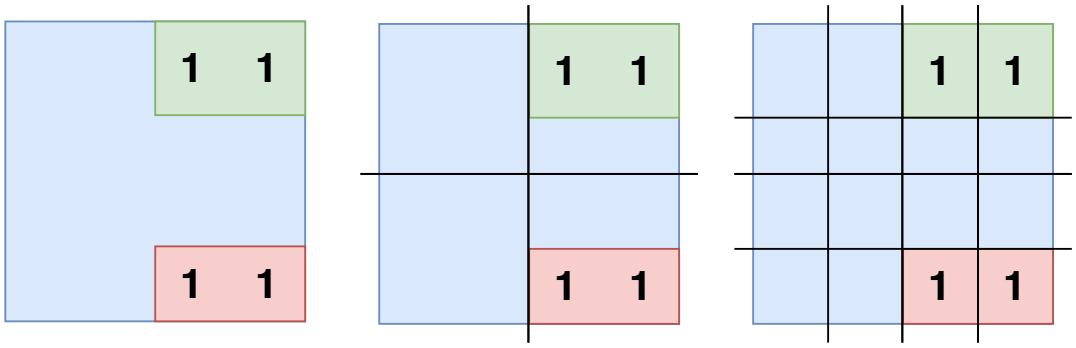
kde N je počet testovaných obrázků, z_i je ground truth počet objektů v $i$tém obrázku a $\hat{z}_i$ je odhadovaný počet objektů v $i$tém obrázku. Čím menší hodnota MAE a MSE, tím lepší je model.

Jednou z dalších možných vyhodnocovacích metrik je *Grid Average Mean Error* ($GAME$), která byla představena v práci *Extremely Overlapping Vehicle Counting* [5]. Tato metrika poskytuje informaci o prostorové chybě, tedy s jakou chybou vyhodnocovaný model dokáže lokalizovat vozidla v obraze. $GAME$ je definována vztahem 5.2.

$$GAME(L) = \frac{1}{N}\sum_{n=1}^{N}(\sum_{l=1}^{4^L}|D^l_{I_n} - D^l_{I^{gt}_n}|), \tag{5.2}$$

kde N je celkový počet snímků, $D^l_{I_n}$ je odhadovaná mapa hustoty vozidel pro snímek n a oblast l. $D^l_{I^{gt}_n}$ odpovídá ground truth mapě hustoty vozidel. Pro specifickou úroveň L, $GAME(L)$ rozdělí obraz na 4^L nepřekrývajících se oblastí. Chyba je pak dána jako suma průměrných absolutních chyb v daných oblastí. To tedy znamená, že $GAME(0)$ odpovídá metrice MAE. Grafické znázornění principu $GAME$ je uvedeno na obrázku 5.1.

Každá síť byla vyhodnocena na datasetu, na kterém byla natrénována. Abych zjistil, zda vytvořené sítě dokáží dobře generalizovat, vyhodnotil jsem je i na datasetech, na kterých nebyly natrénovány. Proto v následujících sekcích nabízím dva typy tabulek výsledků. V první tabulce vyhodnocuji výsledky modelů, které byly natrénovány a otestovány na jednom datasetu. Výsledky ve druhé tabulce byly vytvořeny pro vyhodnocení schopnosti generalizace. Vyskytují se v ní výsledky modelů, jež byly natrénovány na jednom datasetu a vyhodnoceny na jiném.



$MAE = GAME(0) = 0$ $GAME(1) = 4$ $GAME(2) = 4$

Obrázek 5.1: Princip metriky $GAME$. Ground truth umístění a počet objektů představuje červená oblast. Zelená oblast představuje odhad. Hodnota MAE ($GAME(0)$) je 0, i když se nepodařilo správně lokalizovat objekty. Na vyšších úrovních $GAME$ se špatná lokalizace projeví penalizací.

5.1 Výsledky modelů na datasetu TRANCOS

Tabulka 5.1 uvádí výsledky všech modelů testovaných a trénovaných na datasetu TRANCOS.

Model	MSE	GAME(0) = MAE	GAME(1)	GAME(2)	GAME(3)
CCNN [12]	X	12,49	16,58	20,02	22,41
CCNN	25,01	19,79	21,44	23,68	24,27
Hydra CNN [12]	X	10,99	13,75	16,69	19,32
Hydra CNN	19,73	14,90	18,37	21,68	22,64
PACNN (D^{e_0})	11,16	7,77	11,44	13,76	14,54
PACNN (D^{e_1})	**9,91**	**6,49**	**8,49**	**9,22**	**10,27**
PACNN (D^{e_2})	10,59	7,20	9,05	9,90	10,80
PACNN bez persp.	10,15	6,85	9,83	11,38	12,16
MCNN	12,65	8,78	10,78	11,28	12,21

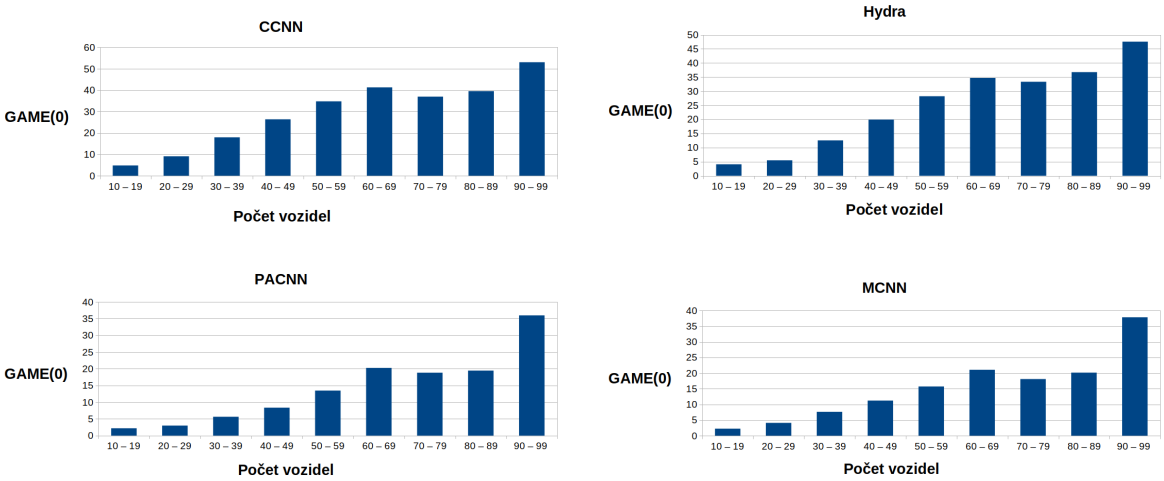
Tabulka 5.1: Výsledky modelů natrénovaných a vyhodnocených na datasetu TRANCOS. Tučně označené hodnoty představují nejlepší dosažený výsledek.

Autoři modelů Counting CNN a Hydra CNN provedli testování na datasetu TRANCOS a proto zde uvádím porovnání mezi jejich výsledky a mými. U Hydry CNN uvádím výsledky natrénovaného modelu se třemi hlavami.

Z uvedených výsledků lze vidět, že v porovnání s původní Counting CNN se mi nepodařilo dosáhnout lepších výsledků. Rozdíl chyb mezi mnou natrénovanou CCNN a původní CCNN při $GAME(0)$ je 7,3, ale při $GAME(3)$ je rozdíl snížen na 1,86. Odhad počtu je tedy sice horší, ale chyba lokalizace by se dala označit za srovnatelnou.

Ačkoliv se mnou natrénovaná Hydra CNN nevyrovnala své původní verzi a ani původní Counting CNN, v porovnání se mou natrénovanou Counting CNN došlo ke zlepšení na všech úrovních metriky $GAME$ i MSE.

Co se týče modelu Perspective–aware CNN, uvedl jsem do tabulky výsledky tří map hustoty před sloučením (D^{e_0}, D^{e_1} a D^{e_2}) a výsledky mapy hustoty po sloučení (v tabulce označeno jako PACNN bez persp.). Na datasetu TRANCOS nelze natrénovat PACNN s vy-



Obrázek 5.2: Změna chyby $GAME(0)$ s rostoucím počtem vozidel v datasetu TRANCOS.

užitím perspektivní mapy, a proto v tabulce nejsou uvedeny její výsledky. Hned na první pohled je vidět, že už samotné mapy před sloučením dávají jedny z nejlepších výsledků a to nejen na nižších úrovních metriky $GAME$, ale i těch vyšších. I v porovnání s CCNN a Hydrou vychází jednotlivé mapy lépe. Zajímavostí je, že odhad mapy D^{e_1} je ve všech kategoriích nejlepší. Po sloučení map do jedné se chyba $GAME(0)$ automaticky nesnížila, ale spíše se vyrovnala lokalizační chyba.

Model MCNN dosahuje podobných výsledků ve všech kategoriích jako PACNN po sloučení map.

Celkově nejlepších výsledků na datasetu TRANCOS dosáhl model PACNN. Velmi podobných výsledků dosáhl i model MCNN. Model Hydra CNN dosahuje lepších výsledků ve všech kategoriích než CCNN, ale horších než ostatní modely. Nejhorších výsledků dosahuje model CCNN.

Grafy na obrázku 5.2 znázorňují, jak se mění chyba s rostoucím počtem vozidel v obraze při $GAME(0)$. Jak je vidět, grafy každého modelu jsou si velmi podobné. Všechny modely zvládají lépe odhadnout menší počet vozidel v obraze. Při nárůstu počtu vozidel se chyba znatelně zhoršuje. Je to způsobeno tím, že v datasetu TRANCOS se vyskytují vozidla těsně u sebe a často se navzájem překrývají, což znesnadňuje správnou predikci.

Tabulka 5.2 uvádí výsledky modelů natrénovaných na datasetech CARPK a PUCPR+. Překvapivě nejlepších výsledků na nižších úrovních $GAME$ dosáhla CCNN natrénovaná na datasetu PUCPR+. Na vyšších úrovních dosahuje nejlepších výsledků PACNN s využitím perspektivní mapy natrénovaná na datasetu PUCPR+. U všech ostatních modelů došlo k výraznému zhoršení výsledků, což bylo očekávatelné. Za zmínku také stojí fakt, že všechny modely natrénované na datasetu PUCPR+ vykazují lepší výsledky, než modely natrénované na CARPK.

U modelu PACNN se mapa perspektivy v tomto případě prokázala jako účinná. U obou natrénovaných modelů dosáhla PACNN s využitím perspektivy lepších výsledků než bez jejího použití, což dokazuje, že mapa perspektivy pomáhá lépe generalizovat.

Model	MSE	GAME(0) = MAE	GAME(1)	GAME(2)	GAME(3)
CCNN (C)	32,86	29,46	30,24	31,27	32,04
CCNN (P+)	**20,32**	**15,92**	**20,85**	24,92	26,36
Hydra CNN (C)	33,36	30,34	30,59	32,20	31,84
Hydra CNN (P)	25,50	21,56	23,78	26,15	27,25
PACNN bez persp. (C)	39,95	36,92	36,92	36,95	36,97
PACNN bez persp. (P+)	27,91	23,96	24,58	25,72	26,67
PACNN s persp. (C, P+)	33,74	30,20	30,21	30,27	30,35
PACNN s persp. (P+)	25,03	20,72	21,79	**23,40**	**24,46**
MCNN (C)	34,32	30,87	31,52	32,31	33,01
MCNN (P+)	26,60	22,35	24,12	26,35	27,55

Tabulka 5.2: Výsledky modelů natrénovaných na datasetech CARPK (C) a PUCPR+ (P+), a vyhodnocených na datasetu TRANCOS. Tučně označené hodnoty představují nejlepší dosažený výsledek.

5.2 Výsledky modelů na datasetu CARPK

Výsledky modelů testovaných na datasetu CARPK uvádí tabulka 5.3. Pro porovnání s jinými existujícími modely přidávám výsledky modelu *Layout Proposal Network* (LPN) [9], jenž byl testován na datasetu CARPK. LPN bylo testováno metrikou MAE a MSE. Nutno podotknout, že výstupem LPN není odhad mapy hustoty, ale počet vozidel.

Model	MSE	GAME(0) = MAE	GAME(1)	GAME(2)	GAME(3)
LPN	36,79	23,80	X	X	X
CCNN	35,63	30,39	34,32	38,24	40,94
Hydra CNN	51,65	45,63	49,84	52,23	56,34
PACNN (D^{e_0})	**10,15**	**8,27**	13,68	16,18	18,37
PACNN (D^{e_1})	14,06	11,01	13,27	15,21	**16,75**
PACNN (D^{e_2})	15,64	11,76	14,18	16,62	18,76
PACNN bez persp.	11,54	9,03	**12,80**	**14,93**	16,89
*PACNN s persp.	20,47	17,54	19,55	22,63	24,71
MCNN	20,56	18,16	23,74	27,32	30,39

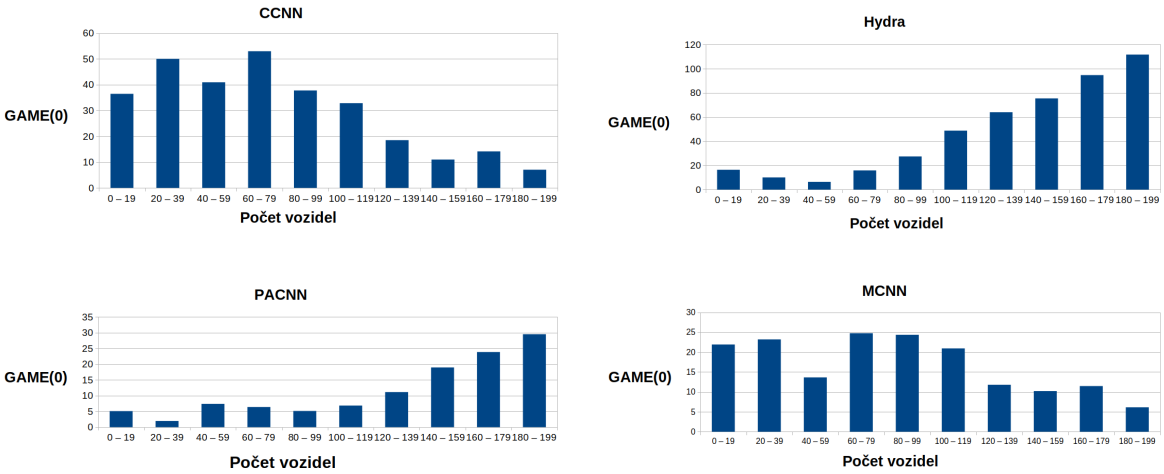
Tabulka 5.3: Výsledky modelů natrénovaných a vyhodnocených na datasetu CARPK. Tučně označené hodnoty představují nejlepší dosažený výsledek.
*PACNN s persp. — model byl trénován na kombinaci datasetů CARPK a PUCPR+.

Modely CCNN a Hydra CNN dopadly na datasetu CARPK nejhůře. Jejich výsledky na všech úrovních metriky $GAME$ se dají označit za nepoužitelné.

Oproti tomu model PACNN znovu dosahuje kvalitních výsledků. Nejnižší hodnoty $GAME(0)$ dosáhla mapa hustoty PACNN D^{e_0}. Tato hodnota je i nejnižší v porovnání se všemi ostatními modely. Nejnižší hodnoty lokalizační chyby $GAME(3)$ dosáhla mapa hustoty PACNN D^{e_1}. Sloučením tří map hustoty D^{e_0}, D^{e_1} a D^{e_2} dosáhla PACNN nejnižší hodnoty $GAME(1)$ a $GAME(2)$. Použití perspektivní mapy výsledky nezlepšilo, naopak je znatelně zhoršilo. PACNN bez perspektivy dosáhla hodnoty $GAME(0)$ 9,03. Po použití perspektivní mapy se chyba zvýšila na 17,54. Chyba se zvyšovala i v dalších úrovních $GAME$.

Výsledky modelu MCNN nebyly tak uspokojivé jako u modelu PACNN, ovšem nebyly ani zdaleka tak špatné, jako u CCNN a Hydry.

Referenční model LPN překonala PACNN, ve všech svých verzích, a MCNN.



Obrázek 5.3: Změna chyby $GAME(0)$ s rostoucím počtem vozidel v datasetu CARPK.

Grafy na obrázku 5.3 zobrazují, jak se mění chyba $GAME(0)$ u měnícího se počtu vozidel v obraze. Grafy jednotlivých modelů si nejsou podobné. U modelů CCNN a MCNN se chyba u menšího počtu vozidel mění nepravidelně. S nárůstem vozidel chyba ale klesá u obou modelů. Oproti tomu modely PACNN a Hydra zvládají lépe predikovat z obrázků s menším počtem vozidel. U modelu PACNN se chyba drží na podobné hladině až po počet vozidel 100–119. Od počtu 120 a výš se chyba znatelně zvyšuje. U Hydry se chyba začíná zvedat od počtu vozidel 100 a víc.

Tabulka 5.4 uvádí výsledky modelů natrénovaných na datasetech TRANCOS a PUCPR+. U modelu CCNN, u obou variant, se výsledky zhoršily natolik, že o jakékoli schopnosti generalizace nelze v tomto případě hovořit. Model Hydra dosáhl velmi podobných výsledků jako při natrénování na datasetu CARPK.

Zajímavých výsledků dosáhl model PACNN bez vužití mapy perspektivy. Ačkoliv se u obou variant výsledky zhoršily, u varianty natrénované na datasetu PUCPR+ se výsledky zhoršily výrazně méně (srovnatelné s PACNN natrénované na CARPK s využitím mapy perspektivy), než u varianty natrénované na datasetu TRANCOS. PACNN natrénovaná na datasetu PUCPR+ s využitím mapy perspektivy dosáhla lehce horších výsledků na vyšších úrovních $GAME$ než bez jejího použití.

Ani jedna z variant modelu MCNN neprokázala schopnost generalizace.

5.3 Výsledky modelů na datasetu PUCPR+

Výsledky modelů trénovaných a testovaných na datasetu PUCPR+ uvádí tabulka 5.5. Znovu uvádím výsledky modelu LPN, jenž byl také testován na datasetu PUCPR+ [9].

Výsledky modelů CCNN a Hydra jsou znovu nepoužitelné. Ačkoliv by měla Hydra být schopnější než CCNN, její výsledky tomu nenapovídají. Ze všech modelů dává nejhorší výsledky. CCNN je na tom trochu lépe, ale i tak jsou její chyby na všech úrovních $GAME$ velké.

PACNN opět dosahuje nejlepších výsledků. Každá jednotlivá mapa překonává ostatní modely. Mapa D^{e_1} dosahuje nejlepších hodnot na všech úrovních $GAME$ v porovnání se všemi ostatními modely. Sjednocením map se chyba zlepšila spíše ve vyšších úrovních

Model	MSE	GAME(0) = MAE	GAME(1)	GAME(2)	GAME(3)
CCNN (T)	146,59	139,95	139,95	140,07	140,18
CCNN (P+)	177,27	175,76	175,76	175,87	175,98
Hydra CNN (T)	53,03	41,26	49,65	55,92	60,36
Hydra CNN (P+)	51,54	43,84	47,54	52,95	57,68
PACNN bez persp. (T)	67,54	62,68	63,13	63,95	66,00
PACNN bez persp. (P+)	27,06	19,92	**23,76**	**26,54**	**29,53**
PACNN s persp. (P+)	**26,65**	**19,31**	24,80	28,04	31,84
MCNN (T)	124,09	97,77	101,50	104,79	107,86
MCNN (P+)	92,45	88,29	88,36	88,65	89,81

Tabulka 5.4: Výsledky modelů natrénovaných na datasetech TRANCOS (T) a PUCPR+ (P+), a vyhodnocených na datasetu CARPK. Tučně označené hodnoty představují nejlepší dosažený výsledek.

Model	MSE	GAME(0) = MAE	GAME(1)	GAME(2)	GAME(3)
LPN	36,73	21,88	X	X	X
CCNN	59,25	51,85	72,13	72,31	75,78
Hydra CNN	84,25	74,15	83,20	85,21	85,51
PACNN (D^{e_0})	14,67	11,69	11,94	12,60	13,40
PACNN (D^{e_1})	3,53	**2,64**	**4,22**	**4,85**	**6,90**
PACNN (D^{e_2})	7,90	5,88	9,09	9,93	10,74
PACNN bez persp.	8,56	6,14	8,49	9,02	10,13
PACNN s persp.	**3,45**	2,86	4,97	6,41	7,96
MCNN	31,90	26,49	28,82	29,95	30,95

Tabulka 5.5: Výsledky modelů natrénovaných a vyhodnocených na datasetu PUCPR+. Tučně označené hodnoty představují nejlepší dosažený výsledek.

$GAME$. Přidáním mapy perspektivy se výsledky znatelně zlepšily oproti sjednoceným mapám bez využití mapy perspektivy. Jejím přidáním se PACNN velmi přiblížila výsledkům mapy D^{e_1}.

Model MCNN sice dosahuje lepších výsledků než CCNN a Hydra, ale ani zdaleka není tak úspěšný jako PACNN.

Referenční model LPN se podařilo překonat pouze modelu PACNN. Model MCNN se výsledkům LPN blížil, rozdíl v chybách MAE byl pouze 4,61.

Porovnání chyb $GAME(0)$ při různých počtech vozidel u tohoto datasetu neuvádím. Dataset PUCPR+ obsahuje pouze 25 testovacích obrázků s rozmezím 1 až 304 vozidel. S takto malým počtem snímků by tyto grafy nebyly směrodatné.

Tabulka 5.6 uvádí výsledky modelů natrénovaných na datasetech CARPK a TRANCOS. U většiny modelů došlo k výraznému zhoršení výsledků. Model PACNN natrénovaný na datasetu TRANCOS bez využití mapy perspektivy zaznamenal méně výrazné zhoršení. Nejvíce překvapivých výsledků dosáhl model MCNN natrénovaný na datasetu TRANCOS, který překonal všechny ostatní modely na všech úrovních $GAME$ i MSE. Na MSE a prvních třech úrovních $GAME$ překonal i MCNN natrénovanou na datasetu PUCPR+.

5.4 Porovnání rychlostí

Tabulka 5.7 porovnává rychlost predikcí jednotlivých modelů. Testování probíhalo na obrázku o rozměrech 720×1280. U každého modelu bylo provedeno dvacet predikcí, jejichž

Model	MSE	GAME(0) = MAE	GAME(1)	GAME(2)	GAME(3)
CCNN (C)	152,17	114,87	121,13	121,66	122,69
CCNN (T)	101,94	84,37	85,71	87,77	88,14
Hydra CNN (C)	146,75	122,31	124,21	125,96	127,02
Hydra CNN (T)	99,55	84,47	87,88	90,90	91,37
PACNN bez persp. (C)	147,55	115,16	115,46	115,73	115,92
PACNN bez persp. (T)	36,27	30,46	48,24	52,69	58,01
MCNN (C)	141,97	108,39	113,43	113,67	114,17
MCNN (T)	**16,88**	**13,83**	**22,33**	**26,92**	**33,04**

Tabulka 5.6: Výsledky modelů natrénovaných na datasetech TRANCOS (T) a CARPK (C), a vyhodnocených na datasetu PUCPR+. Tučně označené hodnoty představují nejlepší dosažený výsledek.

časy byly zprůměrovány. U modelů CCNN a Hydra CNN bylo provedeno testování rychlostí jak na jednom vzorku z obrázku, tak i na celém obrázku. U testování celého obrázku byla využita metoda posuvného okénka s posunem 10 pixelů. Testování probíhalo na procesoru Intel Core i5–4690 a na grafické kartě NVIDIA GeForce GTX 970.

Model CCNN prokázal, že dokáže velmi rychle predikovat malé vzorky z obrázku. Za jednu sekundu by dokázal predikovat až 197 vzorků o velikostech 72×72. U modelu Hydra CNN došlo v porovnání s CCNN k velkému zpomalení, způsobeném mnohem složitější architekturou. Rychlost predikcí na celém obrázku u obou modelů je velmi pomalá. Je to především způsobeno zvoleným krokem posuvného okénka, který je velmi malý. Z celého obrázku bylo extrahováno celkem 7 020 vzorků, které modelu CCNN trvalo zpracovat přes 3 sekundy a modelu Hydra přes 1 minutu. Zrychlení by tedy bylo možné s nastavením většího kroku, což by ale mohlo mít za následek ztrátu přesnosti.

Použití mapy perspektivy u modelu PACNN se projevilo jako menší zpoždění. Bez využití mapy perspektivy dosahoval model PACNN 6,28 FPS[1]. S použitím mapy perspektivy se rychlost propadla na 5,77 FPS.

Modelu MCNN trval zpracovat jeden obrázek 47,65 ms (20,99 FPS), což z něj dělá nejrychlejší model.

Model	Vzorek obrázku (ms)	Celý obrázek (ms)
CCNN	5,06	3 241,23
Hydra CNN	70,0	68 165,42
PACNN bez persp.	X	159,31
PACNN s persp.	X	173,34
MCNN	X	47,65

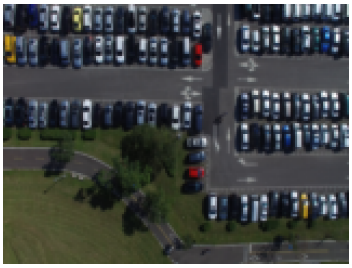
Tabulka 5.7: Porovnání rychlostí predikce. Architektura modelů CCNN a Hydra CNN je patch–based (viz 3) a proto u nich uvádím i rychlost predikce na jednom vzorku z obrázku o velikosti 72×72. Celý obrázek měl velikost 720×1280.

5.5 Porovnání map hustot

Zde nabízím porovnání vygenerovaných odhadů map hustot. První sloupec porovnává mapy z datasetu TRANCOS, druhý sloupec z datasetu CARPK a třetí z datasetu PUCPR+.

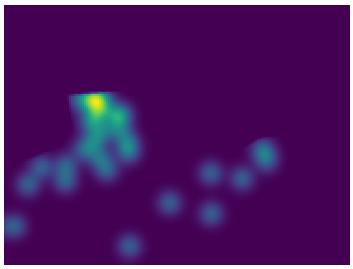
[1] Frames Per Second — obrázků za vteřinu

Snímek z datasetu TRANCOS.



Snímek z datasetu CARPK.

Snímek z datasetu PUCPR+.

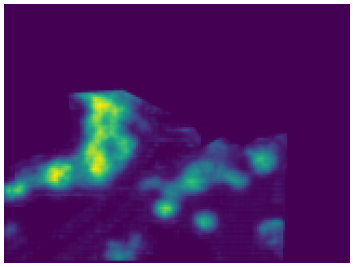


Počet vozidel: 27,64

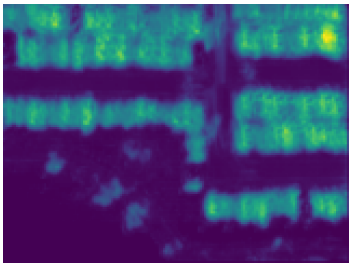
Počet vozidel: 119,00

Počet vozidel: 111,00

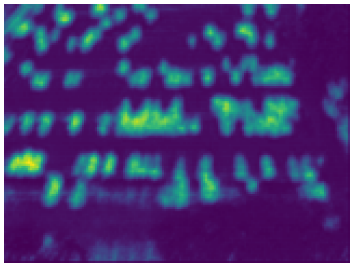
Obrázek 5.4: Ground truth mapy hustoty.



Odhad počtu vozidel: 22,13

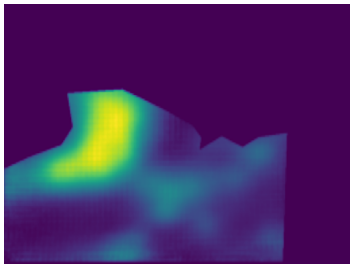


Odhad počtu vozidel: 123,19

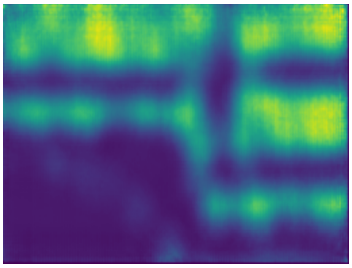


Odhad počtu vozidel: 100,18

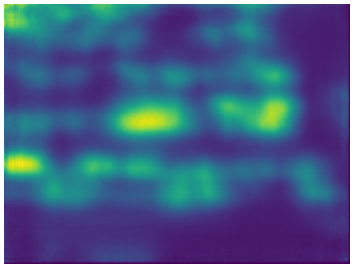
Obrázek 5.5: Odhady map hustot vytvořené modelem Counting CNN.



Odhad počtu vozidel: 45,42

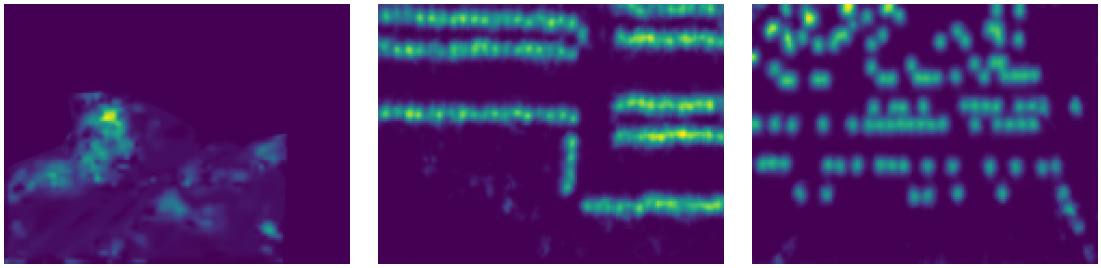


Odhad počtu vozidel: 57,85



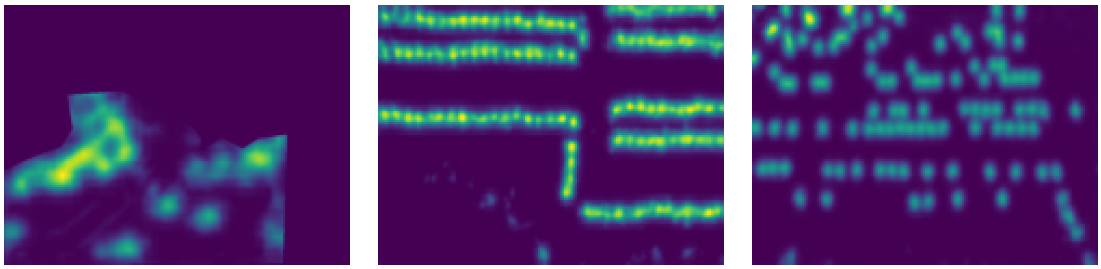
Odhad počtu vozidel: 76,77

Obrázek 5.6: Odhady map hustot vytvořené modelem Hydra CNN.



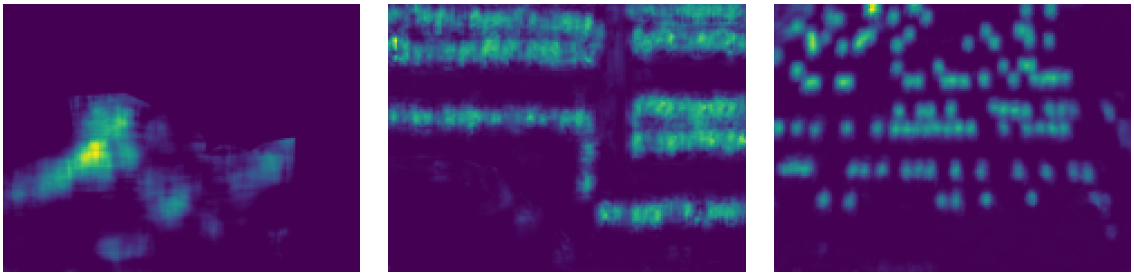
Odhad počtu vozidel: 27,51 | Odhad počtu vozidel: 114,79 | Odhad počtu vozidel: 108,64

Obrázek 5.7: Odhady map hustot vytvořené modelem Perspective–Aware CNN bez využití mapy perspektivy.



*Odhad počtu vozidel: 24,70 | Odhad počtu vozidel: 98,24 | Odhad počtu vozidel: 109,55

Obrázek 5.8: Odhady map hustot vytvořené modelem Perspective–Aware CNN s využitím mapy perspektivy. * — model byl trénován na datasetu PUCPR+.



Odhad počtu vozidel: 30,96 | Odhad počtu vozidel: 127,48 | Odhad počtu vozidel: 94,18

Obrázek 5.9: Odhady map hustot vytvořené modelem Multi–column CNN.

Kapitola 6

Závěr

Hlavním cílem mojí bakalářské práce bylo nastudovat moderní přístupy k počítání vozidel v obraze, otestovat je a porovnat. Zaměřil jsem se na odhad počtu za pomocí mapy hustoty. Celkem jsem si vybral čtyři modely konvolučních neuronových sítí, které jsem natrénoval a otestoval na třech různých datasetech (TRANCOS, CARPK a PUCPR+).

Prvním vyzkoušeným modelem byla Counting CNN. CCNN dosáhla uspokojivých výsledků pouze na datasetu TRANCOS. Jako další nevýhodou tohoto modelu se projevila její rychlost. Predikce jednoho obrázku o rozlišení 720×1280 trvala průměrně 3,24 sekund.

Dalším otestovaným modelem byla Hydra CNN, jejíž architektura byla scale–aware, a která z části využívala modelu Counting CNN. Nejlepších výsledků dosáhla u datasetu TRANCOS, kde překonala CCNN. Na ostatních datasetech dopadla Hydra neuspokojivě. Rychlost predikce byla také jedním z nedostatků. Jeden obrázek o rozlišení 720×1280 Hydra zpracovávala průměrně 68,17 sekund.

Jako významným zlepšením se projevil model Perspective–Aware CNN, který ke zpřesnění svých predikcích využil mapy perspektivy. PACNN byla testována jak s využitím mapy perspektivy, tak i bez ní. PACNN bez perspektivy dosáhla na všech datasetech vynikajících výsledků a překonala i referenční modely. Při použití mapy perspektivy se výsledky zlepšily na datasetu PUCPR+, ale na datasetu CARPK zhoršily. Uplatnění našla mapa perspektivy spíše při testování schopnosti generalizace, kde u datasetu TRANCOS zlepšila výsledky, v porovnání s PACNN bez použití mapy perspektivy, obou modelů natrénovaných na datasetech CARPK a PUCPR+. Rychlost predikce byla také slušná. Jeden obrázek o rozlišení 720×1280 PACNN bez využití mapy perspektivy zpracovávala průměrně 0,159 sekund a s využitím 0,173 sekund.

Posledním otestovaným modelem byla MCNN. Výsledky této konvoluční neuronové sítě byly spíše průměrné. Na datasetu TRANCOS dosáhla podobných výsledků jako PACNN a překonala i referenční modely. Na datasetu CARPK sice dokázala MCNN překonat referenční model, ale oproti PACNN nebyly výsledky tak uspokojivé. Na datasetu PUCPR+ překonala pouze CCNN a Hydru. Velkou výhodou modelu MCNN se stala jeho rychlost predikce. Jeden obrázek mu trvalo zpracovat průměrně 0,048 sekund, čímž se stal nejrychlejším natrénovaným modelem.

Z dosažených výsledků vyšel nejlépe model Perspective–Aware CNN. V budoucnu bych další experimenty cílil právě na tento model a pokusil bych se zlepšit využití perspektivní mapy např. využitím jiného datasetu, který obsahuje více snímků s různou perspektivou. Model MCNN by také stál za dodatečné experimenty.

Literatura

[1] Bishop, C. M. *Pattern Recognition and Machine Learning (Information Science and Statistics)*. Springer-Verlag, 2006. ISBN 0387310738.

[2] Boominathan, L., Kruthiventi, S. a Babu, R. CrowdNet: A Deep Convolutional Network for Dense Crowd Counting. In:. říjen 2016, s. 640–644.

[3] Fu, M., Xu, P., Li, X. et al. Fast crowd density estimation with convolutional neural networks. *Engineering Applications of Artificial Intelligence*. Srpen 2015, roč. 43.

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

[5] Guerrero Gómez Olmedo, R., Torre Jiménez, B., López Sastre, R. et al. Extremely Overlapping Vehicle Counting. In: *Iberian Conference on Pattern Recognition and Image Analysis (IbPRIA)*. 2015.

[6] Krizhevsky, A., Sutskever, I. a Hinton, G. E. ImageNet Classification with Deep Convolutional Neural Networks. In: *Advances in Neural Information Processing Systems 25*. Curran Associates, Inc., 2012, s. 1097–1105. Dostupné z: http://papers.nips.cc/paper/4824-imagenet-classification-with-deep-convolutional-neural-networks.pdf.

[7] Lempitsky, V. a Zisserman, A. Learning To Count Objects in Images. In: *Advances in Neural Information Processing Systems 23*. Curran Associates, Inc., 2010, s. 1324–1332. Dostupné z: http://papers.nips.cc/paper/4043-learning-to-count-objects-in-images.pdf.

[8] Loy, C. C., Chen, K., Gong, S. a Xiang, T. Crowd Counting and Profiling: Methodology and Evaluation. In: *Modeling, Simulation and Visual Analysis of Crowds: A Multidisciplinary Perspective*. New York, NY: Springer New York, 2013, s. 347–382. Dostupné z: https://doi.org/10.1007/978-1-4614-8483-7_14. ISBN 978-1-4614-8483-7.

[9] Meng Ru, H., Yen Liang, L. a Winston, H. H. Drone-based Object Counting by Spatially Regularized Regional Proposal Networks. In: IEEE. *The IEEE International Conference on Computer Vision (ICCV)*. 2017.

[10] Miaojing, S., Zhaohui, Y., Chao, X. et al. Perspective-Aware CNN For Crowd Counting. *CoRR*. 2018, abs/1807.01989. Dostupné z: http://arxiv.org/abs/1807.01989.

[11] Nwankpa, C., Ijomah, C., Gachagan, A. et al. Activation Functions: Comparison of trends in Practice and Research for Deep Learning. *CoRR*. 2018, abs/1811.03378. Dostupné z: http://arxiv.org/abs/1811.03378.

[12] Oñoro, D. a López Sastre, R. Towards Perspective-Free Object Counting with Deep Learning. In: *Computer Vision – ECCV 2016*. Springer International Publishing, 2016, s. 615–629. ISBN 978-3-319-46478-7.

[13] Pham, V.-Q., Kozakaya, T., Yamaguchi, O. et al. COUNT Forest: CO-voting Uncertain Number of Targets using Random Forest for Crowd Density Estimation. In:. Prosinec 2015.

[14] Sharma, S. *Activation Functions in Neural Networks* [online]. Září 2017 [cit. 2020-5-2]. Dostupné z: https://towardsdatascience.com/activation-functions-neural-networks-1cbd9f8d91d6.

[15] Simonyan, K. a Zisserman, A. Very Deep Convolutional Networks for Large-Scale Image Recognition. *ArXiv 1409.1556*. Září 2014.

[16] Skaug, C., Ai, H. a Bai, B. End-to-end crowd counting via joint learning local and global count. In:. Září 2016, s. 1215–1219.

[17] Soni, D. *Supervised vs. Unsupervised Learning* [online]. Březen 2018 [cit. 2020-5-7]. Dostupné z: https://towardsdatascience.com/supervised-vs-unsupervised-learning-14f68e32ea8d.

[18] Vishwanath, S. a Vishal, M. P. A Survey of Recent Advances in CNN-based Single Image Crowd Counting and Density Estimation. *CoRR*. 2017, abs/1707.01202. Dostupné z: http://arxiv.org/abs/1707.01202.

[19] Wang, C., Zhang, H., Yang, L. et al. Deep People Counting in Extremely Dense Crowds. In: *Proceedings of the 23rd ACM International Conference on Multimedia*. New York, NY, USA: Association for Computing Machinery, 2015, s. 1299–1302. Dostupné z: https://doi.org/10.1145/2733373.2806337. ISBN 9781450334594.

[20] Zhang, C., Li, H., Wang, X. et al. Cross-scene crowd counting via deep convolutional neural networks. In: *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 2015, s. 833–841.

[21] Zhang, Y., Zhou, D., Chen, S. et al. Single-Image Crowd Counting via Multi-Column Convolutional Neural Network. In: *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 2016, s. 589–597.

[22] Zhao, Z., Li, H., Zhao, R. et al. Crossing-Line Crowd Counting with Two-Phase Deep Neural Networks. In:. říjen 2016, s. 712–726. ISBN 978-3-319-46483-1.

[23] Zhou, W., Bovik, A. C., Sheikh, H. R. et al. Image quality assessment: from error visibility to structural similarity. *IEEE Transactions on Image Processing*. 2004, roč. 13, č. 4, s. 600–612.