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 **FILIP VÁGNER**
AUTHOR

VEDOUCÍ PRÁCE **Ing. JAKUB ŠPAŇHEL**
SUPERVISOR

BRNO 2022

Vysoké učení technické v Brně
Fakulta informačních technologií

Ústav počítačové grafiky a multimédií (UPGM) Akademický rok 2021/2022

Zadání bakalářské práce

Student: **Vágner Filip**
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 se zaměřením na rozdílné velikosti objektů ve scéně, implementujte je s pomocí dostupných toolkitů a experimentujte s nimi.
4. Získejte 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:

- DOBEŠ Petr, ŠPAŇHEL Jakub, BARTL Vojtěch, JURÁNEK Roman a HEROUT Adam. Density-Based Vehicle Counting with Unsupervised Scale Selection. In: *Digital Image Computing: Techniques and Applications 2020*. Melbourne: Institute of Electrical and Electronics Engineers, 2020, s. 1-8.
- Dále dle pokynů vedoucího.

Pro udělení zápočtu za první semestr je požadováno:

- Splnění prvních dvou bodů zadaní.
- Rozpracovaný body 3 a 4.

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 2021
Datum odevzdání: 11. května 2022
Datum schválení: 1. listopadu 2021

Abstrakt

Úkolem této práce je porovnání modelů konvolučních neuronových sítí určených k počítání vozidel ve statickém obraze pomocí odhadu hustoty se zaměřením na rozdílné velikosti objektů ve scéně. Celkem byly vyhodnoceny čtyři modely – Scale Pyramid Network, Scale-adaptive CNN, Multi-scale fusion network a CASA-Crowd. Vyhodnocení proběhlo na třech datových sadách – TRANCOS, CARPK, PUCPR+. Nejlepších výsledků dosáhl model Scale Pyramid Network. Na datové sadě TRANCOS dosáhl v metrice Mean Absolute Error hodnoty 5,44 a v metrice GAME(3) hodnoty 9,95.

Abstract

The goal of this work is to compare models of convolutional neural networks designed to count vehicles in a static image using density estimation with a focus on different sizes of objects in the scene. A total of four models were evaluated – Scale Pyramid Network, Scale-adaptive CNN, Multi-scale fusion network and CASA-Crowd. The evaluation was done on three data sets – TRANCOS, CARPK, PUCPR+. Scale Pyramid Network achieved the best results. The model reached 5.44 in the Mean Absolute Error metric and 9.95 in the GAME(3) metric on TRANCOS dataset.

Klíčová slova

konvoluční neuronové sítě, počítání vozidel, odhad hustoty, mapa hustoty, Scale Pyramid Network, Scale-adaptive CNN, Multi-scale fusion network, CASA-Crowd

Keywords

convolutional neural networks, vehicle counting, density estimation, density map, Scale Pyramid Network, Scale-adaptive CNN, Multi-scale fusion network, CASA-Crowd

Citace

VÁGNER, Filip. *Počítání vozidel ve statickém obraze*. Brno, 2022. 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.

.....
Filip Vágner
10. května 2022

Poděkování

Chtěl bych poděkovat panu Ing. Jakubovi Špaňhelovi za jeho odborné vedení, čas, ochotu a cenné rady při psaní této práce.

Obsah

Kapitola 1

Úvod

Tato bakalářská práce se zabývá problémem počítání vozidel ve statickém obraze. K řešení tohoto problému existuje mnoho způsobů, nicméně v poslední době se díky velkému pokroku v oblasti hlubokého učení využívají konvoluční neuronové sítě. V této práci se věnuji neuronovým sítím, které fungují na principu odhadu hustoty. Takovéto sítě produkují na výstupu mapu hustoty, díky které můžeme určit jak počet objektů ve scéně, tak i přibližnou polohu objektů v obraze.

Počítání objektů v obraze není jednoduchý proces, jelikož tyto objekty mohou mít různou velikost a mohou být v záběru pod různými úhly. Proto jsem pro experimenty vybral čtyři modely, které se snaží tyto problémy řešit – Scale Pyramid Network, Scale-adaptive CNN, CASA-Crowd a Multi-scale fusion network. Tyto modely jsou blíže popsány v kapitole 3, kde také popisuji různé metody pro počítání vozidel.

V kapitole 2 se věnuji historii a teorii neuronových sítí s bližším zaměřením na konvoluční neuronové sítě, jejich principy a existující architektury. Následující kapitola 4 se zabývá popisem použitých knihoven, výběrem datových sad a procesem trénování a vyhodnocování vybraných modelů. Kapitola 5 se věnuje dosaženým výsledkům, popisuje metriky použité při vyhodnocování a porovnává natrénované modely. V závěrečné kapitole 6 je zhodnocení této práce společně s rekapitulací dosažených výsledků.

Kapitola 2

Umělé neuronové sítě

2.1 Historie

Počátek neuronových sítí se datuje až k roku 1943, kdy neurofyziolog Warren McCulloch a matematik Walter Pitts vytvořili práci zabývající se fungováním neuronů v mozku. K tomuto účelu vytvořili první jednoduchý matematický model neuronu, který představoval základní část nervové soustavy. Tento model měl pouze několik bipolárních parametrů. Podařilo se jim dokázat, že nejjednodušší typy modelů mohou počítat s jakoukoliv logickou nebo aritmetickou funkcí [30].

V roce 1949 napsal Donald Hebb knihu *The Organization of Behavior*, ve které poukázal na fakt, že nervová spojení jsou posilována pokaždé, co se používají. Navrhl tak učící pravidlo pro synapse neuronů. Tato práce ovlivnila velkou řadu dalších vědců [30]. Následně v roce 1951 vznikl první neuropočítač s názvem *Snark*, který byl vytvořen Marvinem Minskym. Tento neuropočítač ale nebyl nikdy prakticky využit. Z technického hlediska však byl úspěšný a dokonce dokázal sám aktualizovat svoje váhy [30]. Psycholog Frank Rosenblatt poté v roce 1957 zobecnil model neuronu a tím vytvořil tzv. *perceptron*. Tento model počítal s parametry v oboru reálných čísel a obsahoval n vstupních a m výstupních neuronů [16]. Dále pro tento model vytvořil učící algoritmus, který po konečném počtu kroků dokázal nalézt odpovídající váhové vektory parametrů pro daná trénovací data [30].

Společně s Charlesem Wightmanem poté v letech 1957 až 1958 sestrojili první neuropočítač s názvem *Mark I Perceptron*. Tento počítač byl navržen pro rozpoznávání znaků. Díky úspěšné prezentaci tohoto počítače se neurovýpočty staly novým předmětem výzkumu. Následkem toho se Frank Rosenblatt dodnes považuje za zakladatele hlubokého učení [30].

V roce 1960 byl představen další zobecněný model neuronu s názvem *ADALINE* (ADAptive LInear NEuron), jehož autorem byl profesor Bernard Widrow. Tento model byl velmi podobný perceptronu, ale využíval nové vylepšené učící pravidlo [30].

Na přelomu 50. a 60. let poté dochází k pokroku v oblasti návrhu a implementace nových modelů neuronových sítí. Většina badatelů však přistupovala k neuronovým sítím pouze z experimentálního hlediska a neřešila analytický výzkum neuronových sítí. Dále byl obor neuronových sítí vyčerpán z hlediska pokroku a bylo nutné objevit nové myšlenky a postupy [30].

Následně od roku 1967 probíhal výzkum neuronových sítí pouze ojediněle. Většina publikací byla publikována pod hlavičkou podobných oborů jako například adaptivní zpracování signálů, rozpoznávání obrazců apod. [30]

Na počátku 80. let se začali objevovat nové projekty zaměřené na vývoj neuropočítačů, které financovala americká agentura DARPA. V roce 1986 badatelé ze skupiny PDP (Paral-

lel Distributed Processing Group) představili nový učící algoritmus zpětného šíření chyby (backpropagation). Tento algoritmus se využívá dodnes [30].

V roce 1987 se konala v San Diegu první velká konference zaměřená na neuronové sítě a byla založena mezinárodní společnost pro výzkum neuronových sítí.

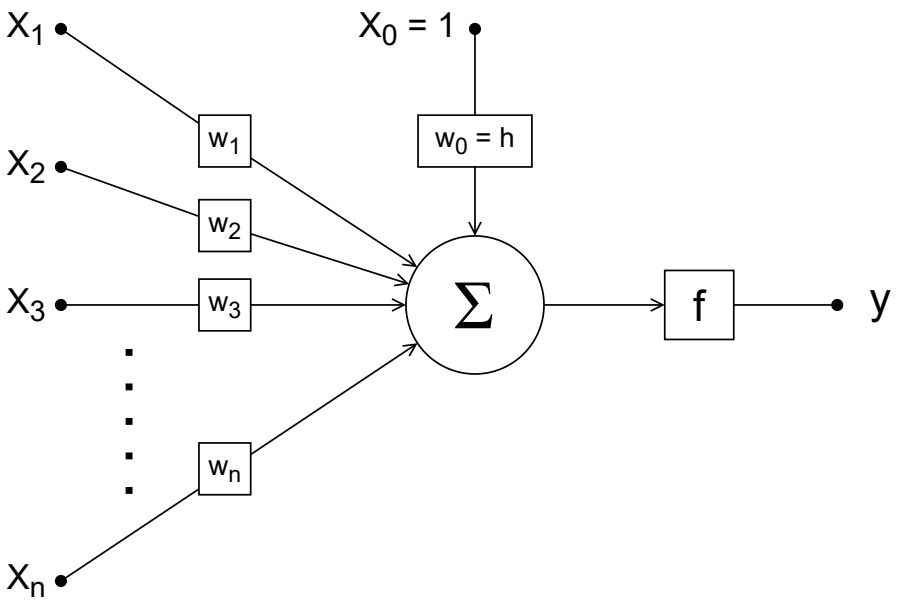
2.2 Architektura

Umělá neuronová síť je výpočetní model skládající se z navzájem propojených formálních neuronů [19]. Tyto formální (umělé) neurony modelují chování skutečných neuronů v centrální nervové soustavě. Formální neuron se také označuje jako *perceptron* a je znázorněn na obrázku 2.1.

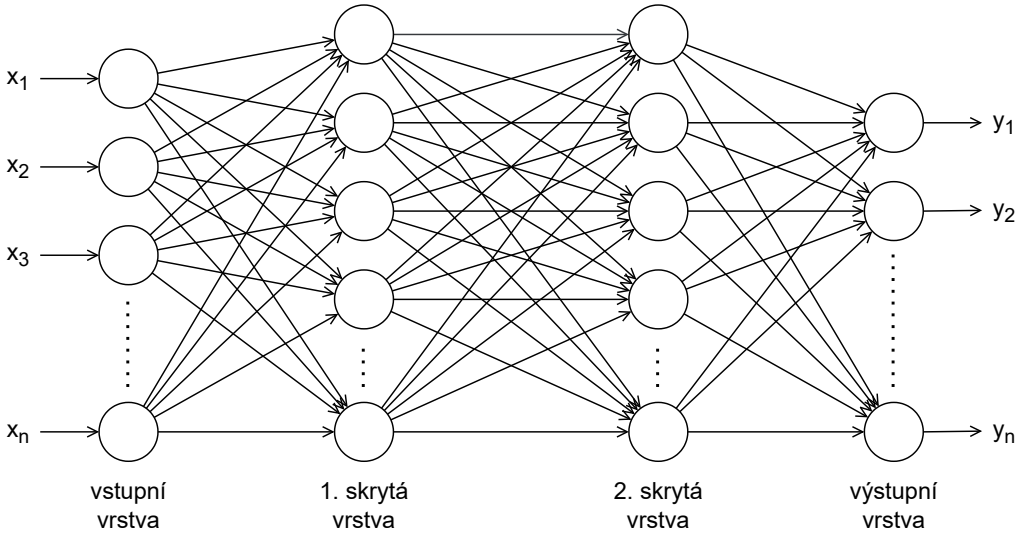
Perceptron má jeden nebo více vstupů x, které jsou ohodnoceny váhou w. Váhy určují propustnost daných vstupů a díky nim je možné perceptron konfigurovat. Všechny vstupy se vynásobí svou váhou a sečtou se dohromady. Výsledek poté přejde do aktivační funkce f, což zajistí nelinearitu výstupu y. Kromě vstupů se také přičítá prahová hodnota (bias) označovaná jako w_0.

Neuronová síť se skládá z vrstev perceptronů zapojených za sebou. Každý perceptron z nižší vrstvy je napojen na všechny perceptrony z vyšší vrstvy. Tyto vrstvy se dělí na vstupní, vnitřní (skryté) a výstupní. Poslední výstupní vrstva má za úkol vracet odhadovaný výsledek. Sítě, které mají více než jednu skrytou vrstvu, jsou tzv. *hluboké neuronové sítě* [7]. Na obrázku 2.2 je uveden příklad neuronové sítě se dvěma skrytými vrstvami.

Neuronové sítě se podle architektury dělí na více typů. Nejznámějšími z nich jsou dopředné neuronové sítě (sekce 2.4), rekurentní neuronové sítě (sekce 2.5) a konvoluční neuronové sítě (sekce 2.6).



Obrázek 2.1: Model perceptronu



Obrázek 2.2: Model neuronové sítě se dvěma skrytými vrstvami

Aktivační funkce

Aktivační funkce je matematická funkce, která transformuje výstup neuronů do určitého oboru hodnot, čímž jsme schopni zajistit nelinearitu. Výběrem správné aktivační funkce můžeme výrazně ovlivnit chování a rychlost učení dané sítě [9]. Existuje mnoho aktivačních funkcí a každá z nich se využívá k různé úloze.

Sigmoid

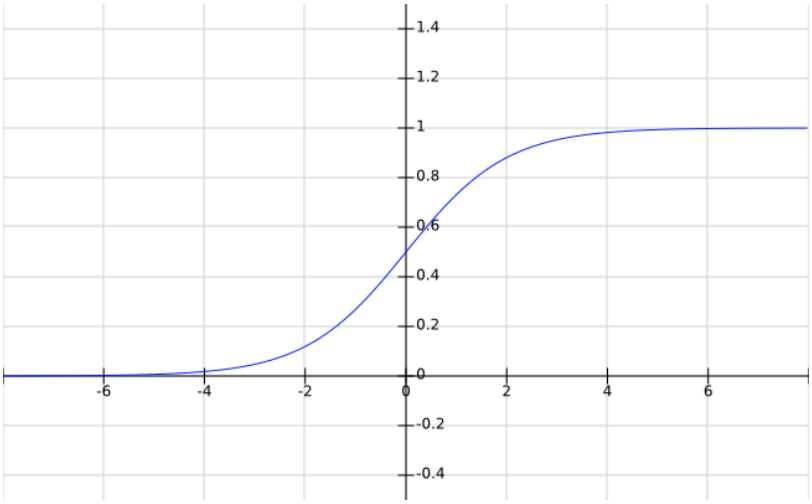
Sigmoid neboli *logistická funkce* je aktivační funkce, která transformuje vstup do rozsahu 0 až 1. Díky tomu se nejčastěji používá při určování pravděpodobnosti. Pokud však jsou vstupní hodnoty daleko od nuly, může se zpomalovat učení sítě a v hlubších sítí může nastat problém mizejícího gradientu [21]. Je definována vztahem 2.1 a její průběh je zobrazen na obrázku 2.3.

$$f(x) = \frac{1}{1 + e^{-x}} \tag{2.1}$$

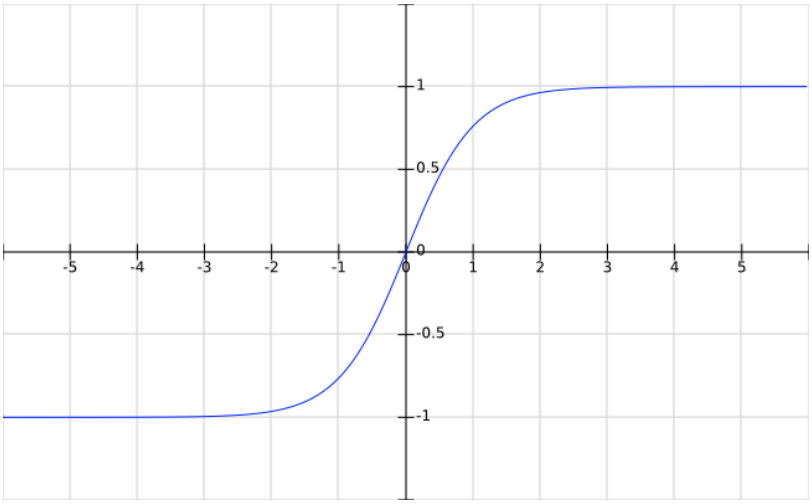
Hyperbolický tangens

Hyperbolický tangens (tanh) je podobná funkci *sigmoid*, ale svůj vstup transformuje do rozsahu -1 až 1. Je také náchylná na problém mizejícího gradientu, ale je preferovanější než funkce sigmoid, jelikož má střed v nule. Používá se nejčastěji při klasifikaci mezi dvěma třídami, a nebo v rekurentních neuronových sítích při rozpoznávání řeči [21]. Je dána vztahem 2.2 a je znázorněna na obrázku 2.4.

$$f(x) = \frac{2}{1 + e^{-2x}} - 1 \tag{2.2}$$



Obrázek 2.3: Průběh aktivační funkce sigmoid



Obrázek 2.4: Průběh aktivační funkce tanh

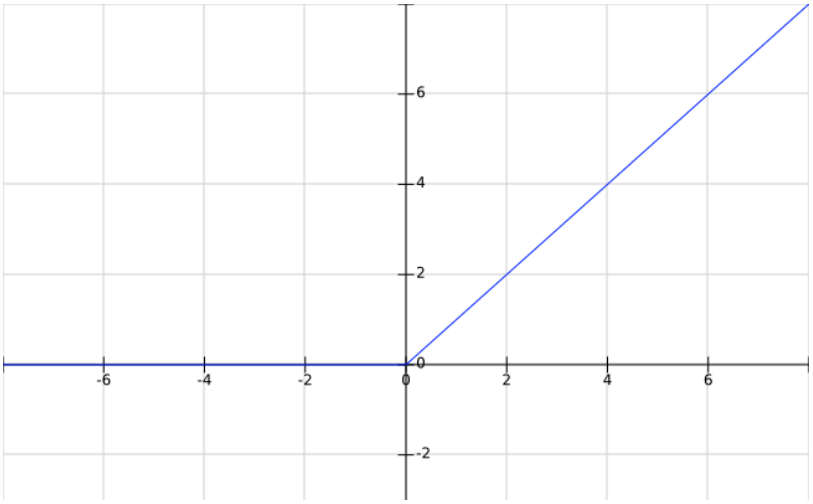
Rectified Linear Unit

Rectified Linear Unit (ReLU) se nejčastěji používá u vícevrstvých konvolučních neuronových sítí. Na rozdíl od funkcí *sigmoid* a *tanh* snižuje dopad mizejícího gradientu, jelikož její gradient je vždy konstantní. Zároveň je méně náročná na výpočet a tudíž se sítě trénují rychleji. Hlavní nevýhodou při její využití je však náchylnost sítě na přeučení. Její průběh je znázorněn na obrázku 2.5 a je dána vztahem 2.3 [21].

$$f(x) = \begin{cases} x & \text{pro } x \geq 0 \\ 0 & \text{pro } x < 0 \end{cases} \tag{2.3}$$

Softmax

Softmax se nejčastěji používá při vícetřídní klasifikaci. Na vstupu očekává vektor reálných hodnot, který poté transformuje na vektor o stejné velikosti, jehož součet je roven 1. Jinými



Obrázek 2.5: Průběh aktivační funkce relu

slovy převede vstupní vektor na vektor rozdělení pravděpodobnosti. Aktivační funkce je dána vztahem 2.4 [21].

$$f(x)_i = \frac{e^{x_i}}{\sum_{j=1}^{K} e^{x_j}} \tag{2.4}$$

2.3 Učení

Aby neuronové sítě dosahovaly co nejlepších výsledků, je důležité, aby měly schopnost učit se. Učení probíhá pomocí úprav vah jednotlivých neuronů. Snažíme se nalézt takové váhy, aby výsledná chyba byla co nejmenší. Učení se většinou dělí na učení s učitelem, učení bez učitele a posilované učení [5].

- **Učení s učitelem** – Neuronové síti předáme kromě vstupních dat také očekávané výstupy (*ground truth* data). Síť poté porovná své výstupy s očekávanými a určí chybu. Podle této chyby se upraví vnitřní nastavení sítě. Tento postup se opakuje tak dlouho, dokud není dosažena námi stanovená minimální chyba.
- **Učení bez učitele** – Síti předáváme pouze vstupní data. Tato data si síť na základě různých vlastností a vztahů sama shlukuje do kategorií.
- **Posilované učení** – Síť se učí pomocí propojování svého prostředí. Využívá k tomu agenta, který se učí ze svých předchozích zkušeností. Výsledky svých akcí si zapamatuje a na základě nich se rozhoduje při dalších akcích.

Učení neboli trénování sítě probíhá tím způsobem, že se snažíme minimalizovat její chybovou funkci [7]. Ze začátku se provede dopředný průchod (forward pass), kde síti předáváme vstupní data, která síť následně zpracuje a vyprodukuje odhadovaný výstup. Tyto výstupy jsou poté porovnány s očekávanými výstupy za pomoci chybové funkce. Výsledek chybové funkce je následně od výstupní vrstvy až po vstupní vrstvu použit k aktualizaci vah jednotlivých neuronů. Tento proces se nazývá zpětná propagace neboli *backpropagation*.

Před trénováním je však nutné vhodně zvolit tzv. *hyperparametry*, které určují chování sítě při trénování. Hyperparametry si síť nemůže určit sama, proto se musí zadat ručně. Mezi hyperparametry patří například:

- velikost dávky – počet vstupních vzorků v jednom průchodu sítí
- počet epoch – počet iterací přes celou datovou sadu
- krok učení – udává, jak moc se mají změnit váhy v reakci na výsledek chybové funkce
- optimalizátor – algoritmus, který mění váhy na základě chybové funkce
- chybová funkce – funkce, která spočítá chybu mezi očekávanými a odhadovanými výstupy (viz 2.3)

Trénovací data většinou rozdělujeme na trénovací a validační. Toto rozdělení slouží k tomu, aby možné kontrolovat chybu na jiných datech, než na kterých byla síť trénována. Díky tomu jsme schopni určit, kdy došlo k přeučení dané sítě.

Chybová funkce

Chybová funkce (loss function) je funkce, která spočítá rozdíl mezi očekávaným výstupem (ground truth) a výstupem odhadovaným neuronovou sítí. Tomuto rozdílu se říká *chyba*. Při učení neuronové sítě se snažíme, aby chyba byla co nejmenší. Příkladem chybové funkce jsou například *mean squared error* (MSE) a *mean absolute error* (MAE), které jsou definovány vztahem 2.5 a 2.6.

$$MSE = \frac{1}{N}\sum_{i=1}^{N}(y_i - \hat{y}_i)^2\,, \tag{2.5}$$

$$MAE = \frac{1}{N}\sum_{i=1}^{N}|y_i - \hat{y}_i|\,, \tag{2.6}$$

kde N je počet trénovacích dat, y_i je ground truth výsledek a $\hat{y}_i$ je výsledek odhadovaný neuronovou sítí.

Zpětné šíření chyby

Zpětné šíření chyby (backpropagation) je algoritmus používaný u neuronových sítí, které fungují na principu učení s učitelem.

Při trénování sítě spočítá chybová funkce rozdíl mezi očekávanými a odhadovanými výstupy. V závislosti na váhách jednotlivých neuronů se vypočítá parciální derivace (gradient) chybové funkce, na základě které se dané váhy aktualizují. Využívá se zde optimalizační algoritmus gradientního sestupu, kdy se po krocích pohybujeme v opačném směru gradientu [28].

Váhy tedy aktualizujeme v opačné hodnotě vypočítané parciální derivace, jelikož se snažíme najít minimální chybu. Používá se zde řetězové pravidlo (chain rule), kdy je část výpočtu gradientu znovu použita pro výpočet následující vrstvy. Aktualizace vah je dána vztahem 2.7.

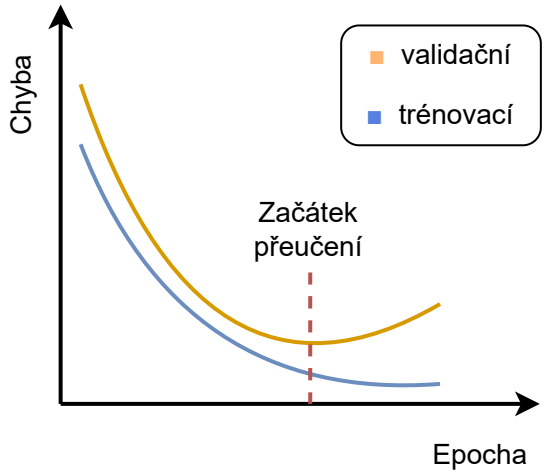
$$a_{n+1} = a_n - \gamma\nabla F(a_n)\,, \tag{2.7}$$

kde a_{n+1} jsou váhy po aktualizaci, a_n jsou váhy před aktualizací, γ je parametr sítě zvaný *learning rate* (krok učení), ∇ je gradient a F chybová funkce.

Nejznámější variantou je *Stochastic gradient descent*, který k výpočtu chybové funkce používá jen jeden náhodně zvolený vzorek trénovacích dat, čímž zajistí rychlejší iterace za cenu nižší míry konvergence.

Přeučení

Přeučení (overfitting) je stav, kdy se síť adaptovala na nežádané vlastnosti trénovací sady a ztrácí tak schopnost generalizace. Většinou nastává při velmi dlouhém učení či při malé velikosti trénovací sady. Přeučení je možné detekovat tak, že se snižuje chyba u trénovacích dat, zatímco u testovacích dat naopak roste. Tento jev je znázorněn na obrázku 2.6.

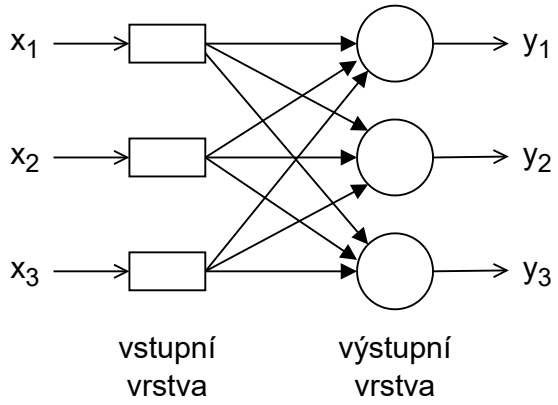


Obrázek 2.6: Ukázka přeučení

2.4 Dopředné neuronové sítě

Jedná se o jeden z prvních a nejjednodušších typů. Spojení mezi neurony v těchto sítích netvoří žádný cyklus. Data jimi prochází pouze jedním směrem (ze vstupní vrstvy do výstupní). Tyto sítě se dále dělí na jednovrstvé a vícevrstvé [25].

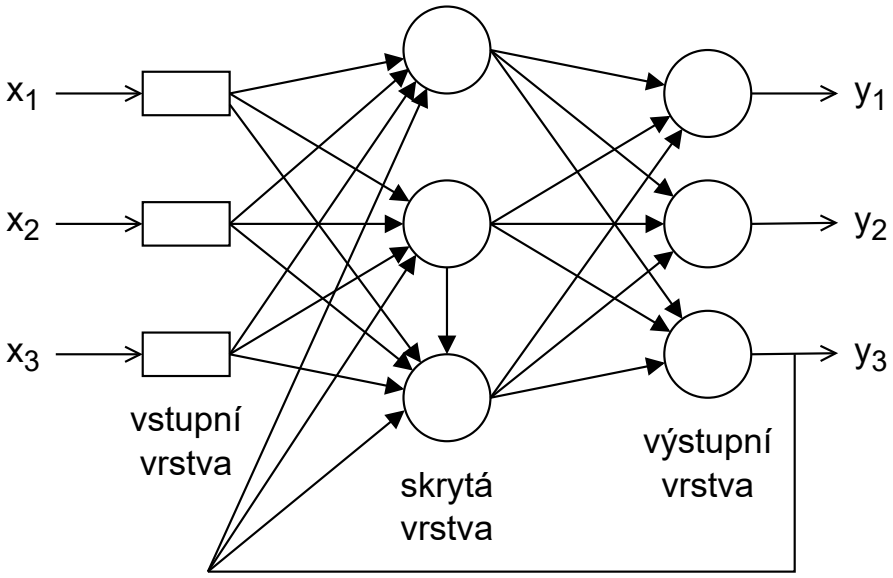
- **Jednovrstvé dopředné neuronové sítě** – Tyto sítě obsahují pouze vstupní vrstvu a výstupní vrstvu. Vstupní vrstva neobsahuje žádné neurony a je přímo propojená s výstupní vrstvou. Využívají se nejčastěji pro lineární filtrování. Ukázku jednovrstvé dopředné neuronové sítě je možno vidět na obrázku 2.7
- **Vícevrstvé dopředné neuronové sítě** – Vícevrstvé dopředné neuronové sítě obsahují jednu nebo více skrytých vrstev mezi vstupní a výstupní vrstvou. Vstupní signály jsou aplikované na první skrytou vrstvu. Výstup každé vrstvy slouží jako vstup pro následující vrstvu. Větší počet skrytých vrstev má za následek zlepšení schopností sítě z hlediska zpracovávání nelineárních závislostí mezi vstupy a výstupy. Příkladem vícevrstvé dopředné neuronové sítě je obrázek 2.2.



Obrázek 2.7: Model jednovrstvé dopředné neuronové sítě

2.5 Rekurentní neuronové sítě

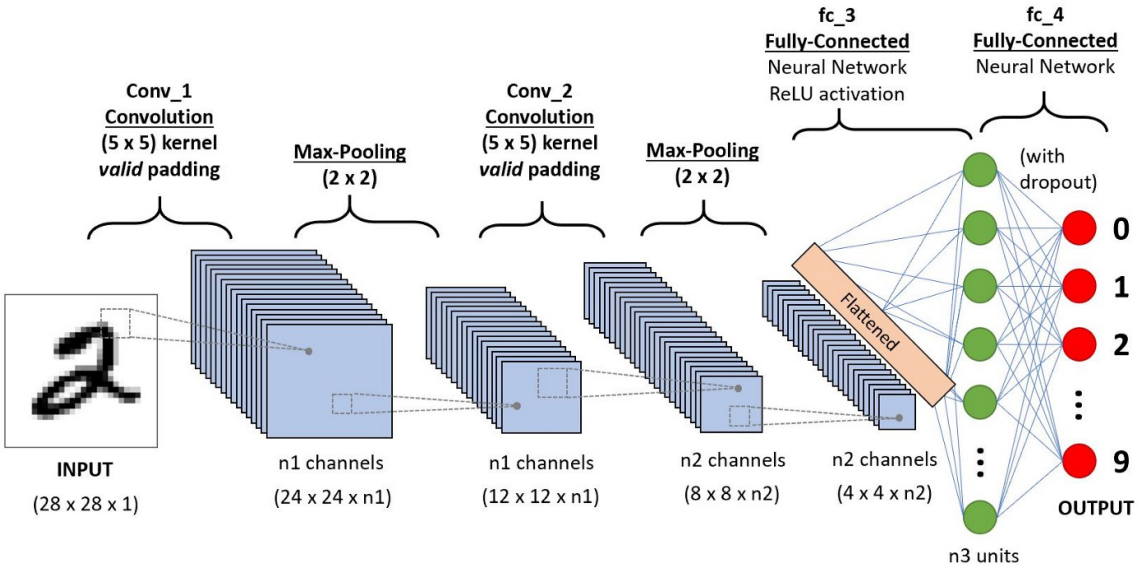
V rekurentních neuronových sítích jsou některé neurony zapojeny do cyklu. Umožňuje to síti uchovávat svůj vnitřní stav. Výstupy neuronů mohou být zapojené do jejich samotných vstupů nebo do vstupů jiných neuronů z předešlých vrstev. Data přenáší jak dopředu, tak i dozadu. Využívají se nejčastěji při předpovídání časových řad nebo při generování textu. Na obrázku 2.8 lze vidět ukázku rekurentní neuronové sítě.



Obrázek 2.8: Model rekurentní neuronové sítě

2.6 Konvoluční neuronové sítě

Konvoluční neuronová síť je třída umělé neuronové sítě nejčastěji používaná v oblasti počítačového vidění, což je oblast zaměřená na získávání a zpracovávání informací z okolního světa. Aby však stroje dokázaly rozpoznat určité objekty, musí se vypořádat s mnohými



Obrázek 2.9: Obecný model sítě pro klasifikaci ručně psaných číslic. Převzato z [20].

překážkami jako jsou perspektiva, změna osvětlení a variabilita objektů. Konvoluční neuronová síť na rozdíl od dopředné neuronové sítě obsahuje navíc *konvoluční* a *pooling* vrstvy. Díky tomu jsou tyto sítě schopné detekovat příznaky ve vstupních datech jako jsou například tvary [7, 4]. Na obrázku 2.9 je vyobrazena ukázka modelu konvoluční neuronové sítě pro klasifikaci ručně psaných číslic. V následujících sekcích popíšu nejpoužívanější vrstvy konvolučních neuronových sítí.

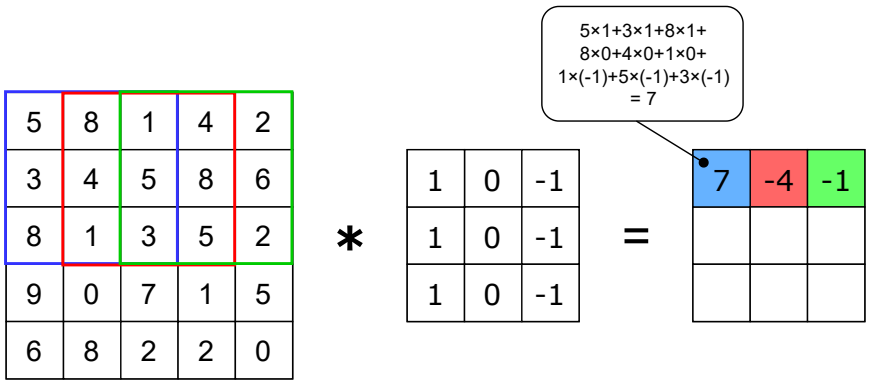
Konvoluční vrstva

Konvoluční vrstva se skládá z filtrů, které se vstupními daty provádí operaci zvanou konvoluce [1]. Při použití je potřeba zadat rozměry (délka, šířka a hloubka) a posun neboli *stride* filtru. Posun filtru udává, jak moc se budeme posouvat po vstupu. Déle je možné zadat *padding*, který způsobí, že vstup rozšíří na takovou velikost, aby po konvoluci měl výstup stejné rozměry jako původní vstup. Pokud se *padding* nepoužije, výstup bude mít oproti vstupu zmenšené rozměry.

Výstupem konvoluční vrstvy je mapa příznaků, což jsou naučené příznaky v prostorových dimenzích (např. hrany). Konvoluční vrstvy využívají velmi často vícero jader. Každé jádro produkuje na výstupu jednu mapu příznaků. Výstupem konvoluční vrstvy je tedy několik map příznaků. Pro dvourozměrná data je konvoluce definována vztahem 2.8.

$$y[m,n] = \sum_{i=0}^{I-1} \sum_{j=0}^{J-1} h[i,j] * x[m-i, n-j]\,, \tag{2.8}$$

kde x udává vstupní obrázek, h konvoluční filtr, y výstupní mapu příznaků, m a n souřadnice pixelu ve vstupním obrázku a I a J rozměry konvolučního filtru. Grafické znázornění konvoluce je na obrázku 2.10.



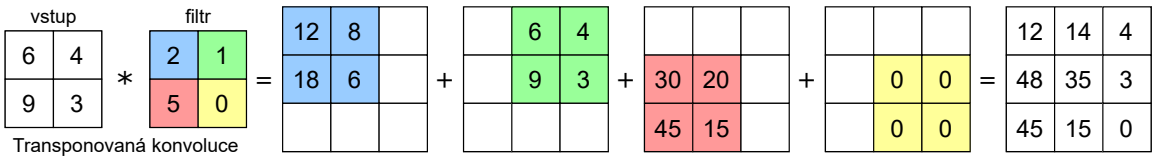
Obrázek 2.10: Ukázka dvourozměrné konvoluce s filtrem o velikosti 3×3 a posuvem 1

Transponovaná konvoluční vrstva

Jedná se o typ konvoluční vrstvy, která je schopna zvětšit rozměry mapy příznaků (upsampling). Tato vrstva bývá často mylně nazývána *dekonvoluční* vrstvou. Stejně jako u konvoluční vrstvy je zde možné nastavit *stride* a *padding*, ale na rozdíl od konvoluční vrstvy fungují jiným způsobem.

Stride určuje, jakým krokem se bude posouvat po výstupu. Na vstupu se posouvá vždy po 1. To má za následek zvětšení rozměrů výstupu, pokud není využit zároveň i padding. Padding u transponované konvoluční vrstvy způsobí, že výstupní mapa bude mít stejné rozměry jako ta vstupní.

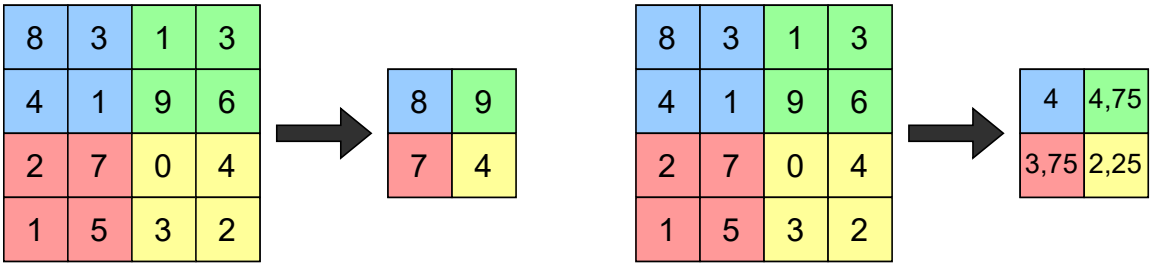
Ukázka transponované konvoluční vrstvy je na obrázku 2.11.



Obrázek 2.11: Ukázka transponované konvoluce s filtrem o velikosti 2×2 a posuvem 1

Pooling vrstva

Pooling se využívá k tomu, aby se redukovala složitost dat pro další vrstvy. Vrstva redukuje množství parametrů v síti tím, že z původní mapy příznaků extrahuje pouze určité části. Využívá k tomu posuvnou matici s definovaným krokem (stride), která přejíždí po vstupních datech a z ní vybere pouze jednu hodnotu, kterou uloží do výstupní mapy příznaků. To má za následek zmenšení rozměrů výstupní mapy příznaků [1]. Nejčastěji se používá *max-pooling*, který z překrývaných dat vybírá největší hodnotu a *average-pooling*, který z vybraných hodnot udělá průměr. Max-pooling a average-pooling jsou znázorněny na obrázku 2.12.



Obrázek 2.12: Ukázka max-pooling (vlevo) a average-pooling (vpravo)

Dávková normalizační vrstva

Dávková normalizační vrstva (batch normalization layer) je vrstva, která normalizuje data, která přes ní prochází. Tímto se snaží redukovat tzv. *internal covariate shift*, což je změna rozložení aktivace neuronů sítě během trénování [14].

Normalizační vrstvou jsme schopni dosáhnout toho, že střední hodnota a rozptyl dat bude vždy stejný, což nám umožňuje rychlejší trénování sítě s možností vyššího trénovacího kroku.

Plně propojená vrstva

Plně propojená vrstva je velmi podobná způsobu, jakému jsou neurony zapojeny v dopředné neuronové síti. Každý neuron je propojen s každým neuronem z předchozí vrstvy a následující vrstvy [1]. Kvůli tomu tato vrstva obsahuje velké množství trénovacích parametrů a její trénování trvá déle.

Tyto vrstvy se používají za posledními konvolučními a pooling vrstvami v síti. Jejich vstupem je vektor a proto se vstupní data musí napřed transformovat do 1D pole. Tato data jsou poté předána do jedné nebo více plně propojených vrstev, které na získané příznaky aplikují své váhy a tím se snaží předpovědět správný výstup sítě, jako je například pravděpodobnost třídy v klasifikačních úlohách.

Dropout vrstva

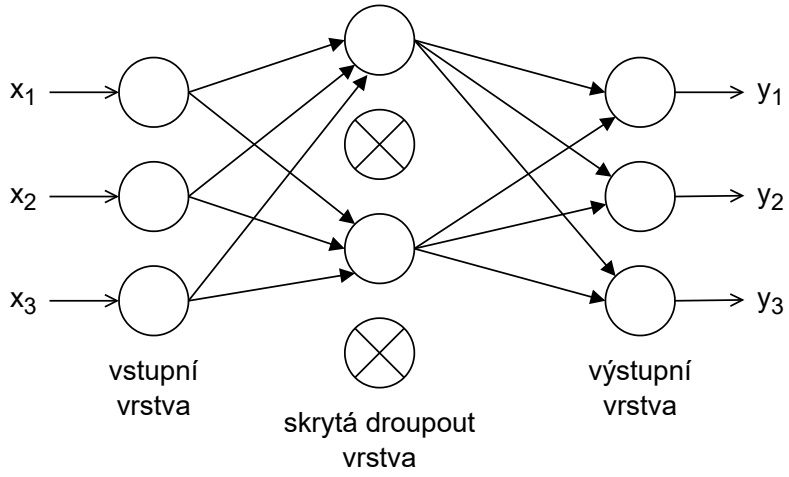
Dropout vrstva je maska, která ignoruje určité množství neuronů podle dané pravděpodobnosti. Zbytek neuronů nechává beze změny. Tato maska je aktivní při trénování sítě, kdy zakryje určitá data, která neprojdou do další vrstvy. Většinou se tato vrstva vkládá mezi plně propojené vrstvy nebo za pooling vrstvy. Dropout se používá převážně k tomu, aby se zabránilo přeučení dané sítě. S jejím použitím probíhá trénování rychleji, ale síť potřebuje více času ke konvergenci. Ukázka dropout vrstvy je vyobrazena na obrázku 2.13.

Architektury konvolučních neuronových sítí

V této sekci představím několik nejznámějších a nejpoužívanějších modelů konvolučních neuronových sítí, jejich architekturu a použití.

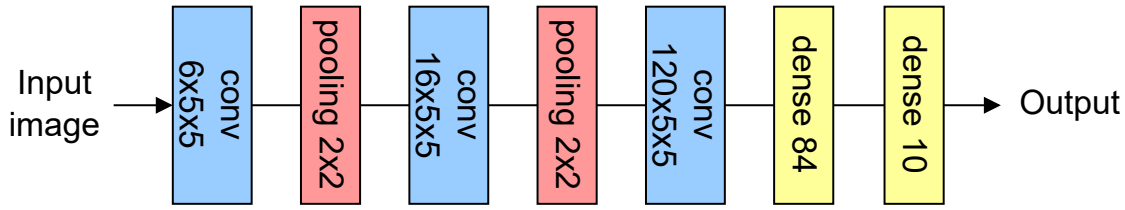
LeNet

LeNet byl vytvořen v roce 1998 a jedná se o jednu z prvních konvolučních neuronových sítí. Pochází z práce *Gradient-Based Learning Applied to Document Recognition* [17]. Byl



Obrázek 2.13: Ukázka dropout vrstvy

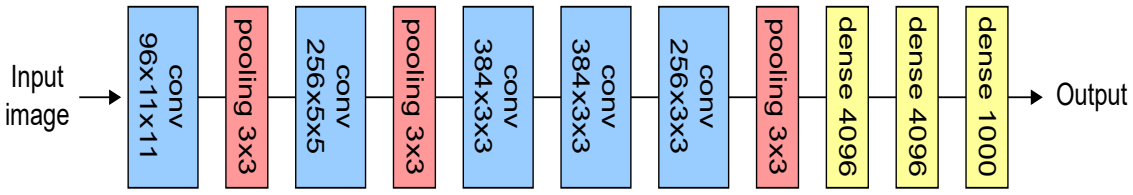
jednou z nejvíce používaných architektur a používá se například pro rozpoznání psaného textu. Hlavním důvodem jeho popularity je jeho jednoduchost a přímočarost. Skládá se ze tří konvolučních vrstev, dvou max-pooling vrstev a dvou plně propojených vrstev. Jeho vstupem je černobílý obrázek o velikosti 32×32 pixelů. Architektura tohoto modelu je znázorněna na obrázku 2.14.



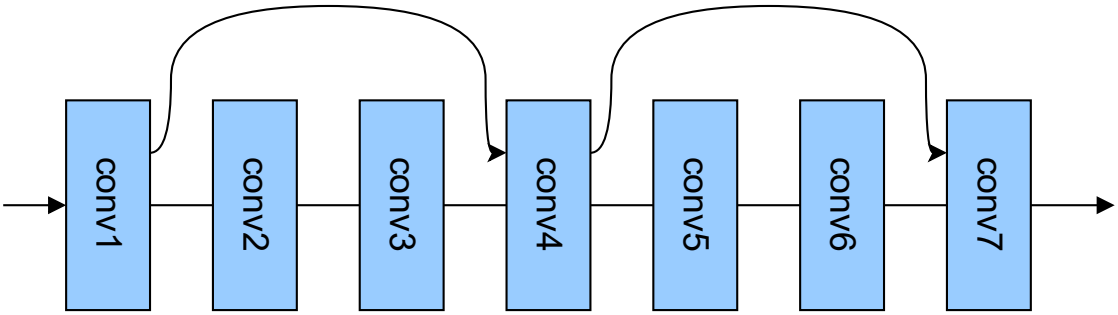
Obrázek 2.14: Ukázka modelu LeNet. Vytvořeno podle [17].

AlexNet

AlexNet je model používaný pro klasifikaci obrazových dat a pochází z práce *ImageNet Classification with Deep Convolutional Neural Networks* [15]. Na rozdíl od modelu LeNet má hlubší architekturu. Skládá se z pěti konvolučních vrstev, tří max-pooling vrstev a tří plně propojených vrstev. Tento model v roce 2012 vyhrál výzvu *ImageNet Large Scale Visual Recognition Challenge* (ILSVRC), kde dosáhl top-5 chyby 15,3 %. Ukázka tohoto modelu je znázorněna na obrázku 2.15



Obrázek 2.15: Ukázka modelu AlexNet. Vytvořeno podle [15].



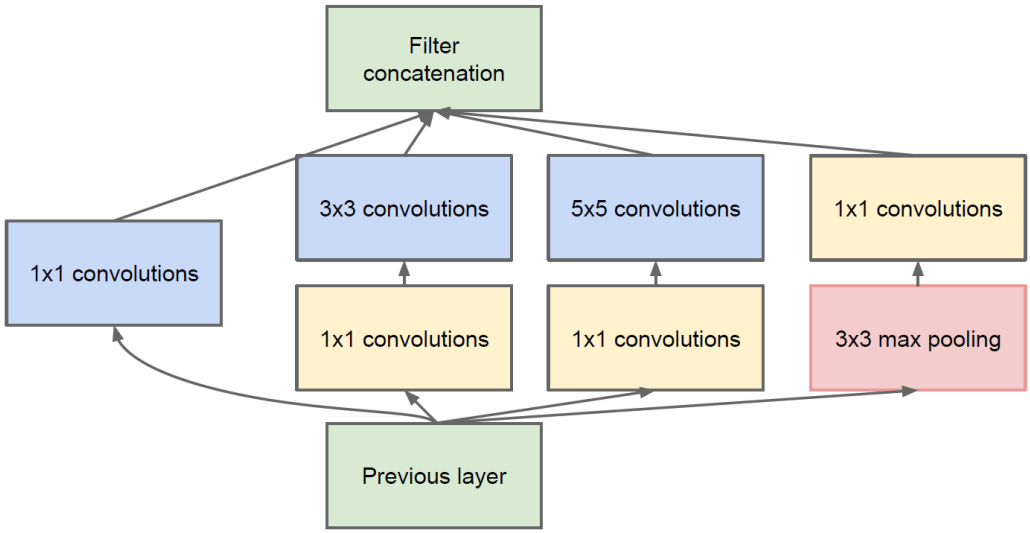
Obrázek 2.16: Ukázka modelu ResNet

ResNet

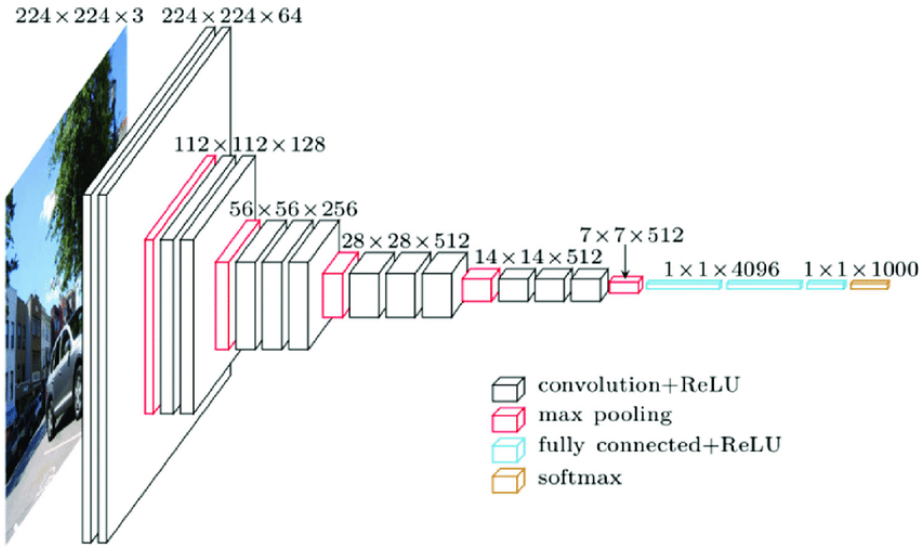
ResNet pochází z práce *Deep Residual Learning for Image Recognition* [10]. Používá princip přeskakovaných spojení, aby přeskočil některé vrstvy. Díky těmto skokům se data mohou vyhnout určitým vrstvám. Tímto se snaží snížit degradaci sítě a také vyřešit problém mizejícího gradientu. Využívá k tomu také vrstvy dávkové normalizace. Většinou se používá skok o dvě nebo tři vrstvy. Jeho architektura je znázorněna na obrázku 2.16

GoogleNet

GoogleNet pochází z práce *Going Deeper with Convolutions* [24]. Skládá se z 22 vrstev a používá tzv. *inception modul*, který je znázorněn na obrázku 2.17. Tento inception modul se skládá z několika konvolučních filtrů různých velikostí a také max-pooling vrstvou o velikosti 3×3. Díky tomu je síť schopna hledat různě velké příznaky ve vstupních datech. Inception modul je však velmi náročný na výpočet, jelikož obsahuje mnoho parametrů. Proto se v něm používají filtry o velikosti 1×1, které snižují počet dimenzí a tím snižují i parametry sítě. Síť obsahuje několik těchto inception modulů zapojených za sebou. GoogleNet vyhrál soutěž ILSVRC 2014 s top-5 chybou 6,67 %.



Obrázek 2.17: Ukázka inception modulu ze sítě GoogleNet. Převzato z [24].



Obrázek 2.18: Ukázka modelu VGG. Převzato z [2].

VGG

Model VGG pochází z práce *Very Deep Convolutional Networks for Large-Scale Image Recognition* [22] a je nástupce modelu AlexNet. Namísto filtrů o rozměrech 11×11 a 5×5 však používá hlubší architekturu s filtry o rozměrech 3×3, jelikož menší filtry vyžadují méně výpočetního výkonu. Používá se ke klasifikaci obrazových dat a jeho vstupem jsou obrázky o velikosti 224×224 pixelů. V soutěži ILSVRC 2014 skončil na druhém místě s top-5 chybou 7,3 %. Ukázka modelu je znázorněna na obrázku 2.18. Model obsahuje 13 konvolučních vrstev, 5 max-pooling vrstev a 3 plně propojené vrstvy.

Kapitola 3

Přístupy k počítání vozidel v obraze

Počítání objektů v obraze je jednou z oblastí počítačového vidění. Její použití může být například při územním plánování, monitorování dopravy, zajištění veřejné bezpečnosti nebo odhadu zalidnění. Nejvíce zaměřovaným tématem v poslední době je počítání lidí v davu (crowd counting), avšak tyto metody lze přenést také na počítání vozidel.

Při počítání objektů se však vyskytuje mnoho problémů. Jedná se například o rozdílnou velikost objektů, nerovnoměrné rozmístění objektů, rozdílné úhly záběru nebo nepříznivé světelné podmínky.

Způsoby počítání objektů se nejčastěji dělí do těchto kategorií [26]:

- **Pomocí detekce** – U této metody se využívá posuvné okénko, které detekuje přítomnost daných objektů ve scéně a jejich počet. K tomuto úkonu je zapotřebí mít dobře natrénovaný klasifikátor. Při vysoké hustotě objektů však velmi často selhává.
- **Pomocí regrese** – Tato metoda dokáže vytvořit přímé mapování mezi vstupním obrázkem a počtem objektů. Využívají k tomu nízkoúrovňové příznaky, které jsou poté pomocí regrese mapovány na počty objektů. Problémem těchto metod je, že nedokáží pracovat s rozdílnou perspektivou a velikostmi objektů ve scéně.
- **Pomocí odhadu hustoty** – Fungují na podobném principu jako způsoby pomocí regrese, ale příznaky si ze vstupních dat dokáží extrahovat samy. Využívají se k tomu konvoluční neuronové sítě, které se snaží naučit mapování lokálních příznaků na jejich odpovídající mapy hustoty.

3.1 Počítání vozidel pomocí odhadu hustoty

Tato práce se zabývá metodou počítání pomocí odhadu hustoty. Výstupem této metody je mapa hustoty, která kromě počtu objektů udává také informaci o jejich rozložení ve scéně. Celkový počet objektů se získá integrací přes celou mapu hustoty.

Využívá se zde učení s učitelem, kdy neuronové síti předáváme očekávané výstupy (ground truth). Vznik těchto ground truth map je popsán v sekci 4.2. V průběhu let vzniklo spoustu modelů konvolučních neuronových sítí, které se snaží problém počítání objektů řešit. Podle [23] se tyto modely dělí dle architektury na:

- **Základní CNN** – Tyto modely používají základní vrstvy konvolučních neuronových sítí. Jedná se o jedny z prvních modelů, které používaly k řešení neuronové sítě. Příkladem těchto modelů je [18].
- **Scale-aware modely** – Jedná se o propracovanější modely, které jsou schopny řešit rozdílné velikosti objektů ve scéně. Využívají k tomu zpravidla vícesloupcové architektury. Mezi tyto sítě patří například [27].
- **Context-aware modely** – Za pomoci lokálních a globálních kontextových informací se snaží dosáhnout nižších chyb v odhadu mapy. Příkladem těchto sítí jsou [11, 13].
- **Multi-task modely** – Kombinují počítání objektů s další úlohou jako například odhad rychlosti davu. Takovéto kombinace mohou síti pomoci k dosáhnutí lepších výsledků. Tímto modelem je například [29].

Dále lze také modely dělit dle způsobu inference:

- **Vzorková inference** – Model se trénuje na malých částech, které jsou vyříznuty z původního snímku. Při vyhodnocování se používá posuvné okénko, které přejíždí po testovacím obrázku a extrahuje části snímku. Tyto části jsou postupně předávány síti, která vytvoří predikce map hustot. Z těchto predikcí je poté agregací sestavena finální mapa hustoty.
- **Celosnímková inference** – Tyto modely jsou nezávislé na velikosti vstupních obrázků a vyhýbají se tak použití posuvného okénka, které je výpočetně náročné. Inference probíhá na celém snímku bez extrahování jeho částí.

V následujících sekcích jsou blíže popsány modely, které jsem zvolil pro reimplementaci a testování.

3.2 Scale Pyramid Network

Scale Pyramid Network (SPN) vychází z práce *Scale Pyramid Network for Crowd Counting* [3]. Tento model využívá jednosloupcovou architekturu s jednotnou velikostí filtrů o velikosti 3×3, která je převzata z předtrénované sítě VGG16 [22].

Do části této sítě je vložen *Scale Pyramid Module* (SPM), který dokáže extrahovat příznaky různých velikostí. Tento Scale Pyramid Module obsahuje čtyři paralelně zapojené vrstvy dilatované konvoluce s parametry *dilation rate* 2, 4, 8 a 12. Každá z těchto vrstev má počet filtrů stejný jako předchozí vrstva, na kterou je SPM napojeno. Na rozdíl od tradiční konvoluce, dilatovaná konvoluce nezvyšuje počet parametrů při zvětšení receptivního pole. Výstupy těchto čtyř vrstev jsou následně sloučeny společně s jejich původním vstupem. Model je díky SPM schopen zajistit přesnější predikce společně s méně náročným trénováním.

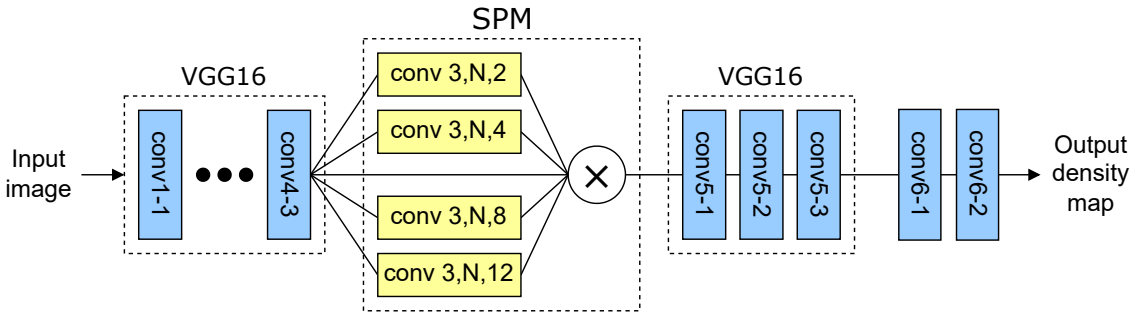
Architektura modelu je znázorněna na obrázku 3.1. Vstupem modelu jsou snímky o velikosti 1/4 původního obrázku. Prvních 10 konvolučních vrstev a 3 max-pooling vrstvy (*conv1-1* až *conv4-3*) jsou převzaty z předtrénované sítě VGG16. Následuje Scale Pyrimad Module, za který se poté napojí vrstvy *conv5-1* až *conv5-3* ze sítě VGG16. Na tyto vrstvy se napojí další konvoluční vrstva s velikostí filtru 3×3 a hloubkou 256. Poslední vrstva má velikost filtru 1×1 s hloubkou 1 a vrací odhadovanou mapu hustoty. Všechny konvoluční vrstvy jsou následovány ReLU aktivační funkcí.

Kvůli použití tří max-pooling vrstev je výsledná mapa hustoty pouze 1/8 velikosti vstupního obrázku. Aby se zachoval počet vozidel po zvětšení mapy, musí dojít k normalizaci. Způsob normalizace je popsán v sekci 4.2.

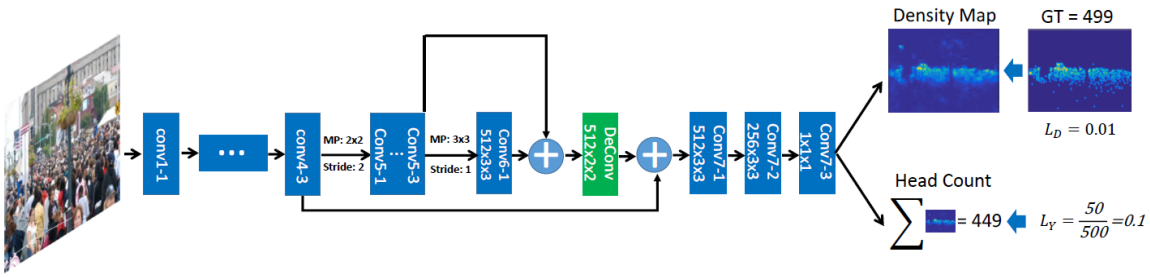
Autoři k trénování použili optimalizační algoritmus *Stochastic gradient descent* a chybovou funkci *Euclidean loss*, která je definována vztahem 3.1.

$$L(\Theta) = \frac{1}{2N} \sum_{i=1}^{N} ||F(X_i; \Theta) - F_i||_2^2 \tag{3.1}$$

kde Θ udává množinu parametrů sítě, X_i značí vstupní obrázek, $F(X_i; \Theta)$ je sítí odhadovaná mapa hustoty a F_i je příslušná ground truth mapa hustoty. N udává počet snímků v trénovací dávce.



Obrázek 3.1: Architektura modelu Scale Pyramid Network. Vytvořeno podle [3].



Obrázek 3.2: Architektura modelu Scale-adaptive CNN. Převzato z [27].

3.3 Scale-adaptive CNN

Tento model vychází z práce *Crowd counting via scale-adaptive convolutional neural network* [27]. Scale-adaptive CNN extrahuje příznaky z vícero vrstev, které následně změní na stejnou velikost. Tyto příznaky se poté zkombinují a vytvoří finální mapu hustoty.

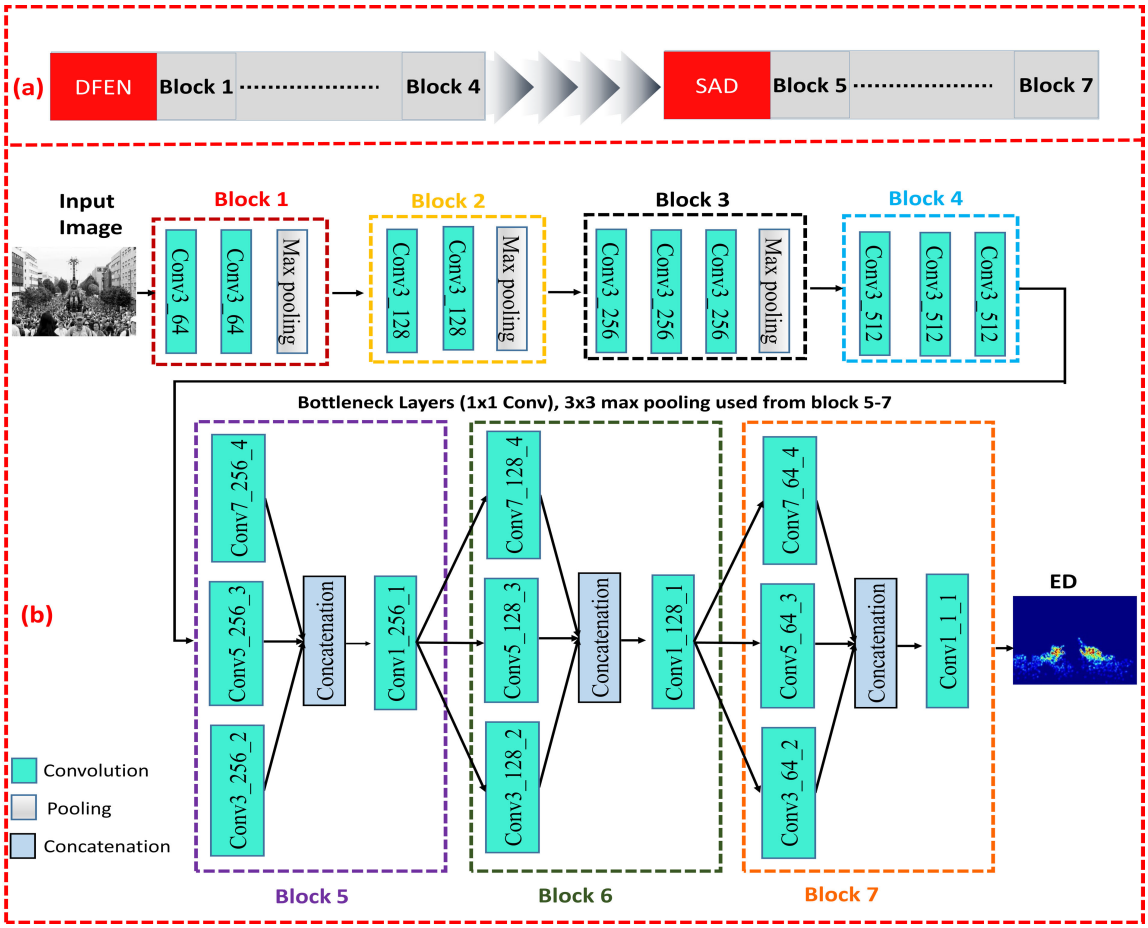
Síť patří do kategorie multi-task modelů, jelikož kromě mapy hustoty vrací také odhadovaný počet objektů ve scéně. K tomuto úkolu používá chybové funkce *Euclidean loss* (pro mapu hustoty) a *Relative count loss* (pro odhadovaný počet), která je definována vztahem 3.2.

$$L_Y(\Theta) = \frac{1}{N} \sum_{i=1}^{N} ||\frac{F_Y(X_i; \Theta) - Y_i}{Y_i + 1}||^2, \tag{3.2}$$

kde $F_Y(X_i;\Theta)$ udává odhadovaný počet objektů, Y_i značí ground truth počet objektů a N je počet snímků v trénovací dávce.

Architektura Scale-adaptive CNN je zobrazena na obrázku 3.2. Model se skládá ze sedmnácti konvolučních vrstev, pěti max-pooling vrstev a jedné transponované konvoluční vrstvy. Vstupem jsou vzorky o velikosti 1/4 původního obrázku. Prvních 10 konvolučních vrstev a 3 max-pooling vrstvy mají architekturu převzatou ze sítě VGG16. Z vrstev *conv4-3* a *conv5-3* jsou vyextrahovány mapy příznaků, které jsou následně použity pro doplnění informací v pozdějších vrstvách. Všechny konvoluční vrstvy mají velikost filtru 3×3 a kromě úplně poslední vrstvy jsou následovány ReLU aktivační funkcí. Transponovaná konvoluční vrstva má velikost filtru 2×2 a má za úkol zvětšit mapu hustoty na 1/8 vstupního vzorku.

Podle autorů modelu lze síť natrénovat dvěma způsoby. Buď lze síť trénovat s oběma výstupy najednou jako multi-task model, anebo nejprve natrénovat pouze s výstupem mapy hustoty a následně po její konvergenci doladit váhy tréninkem s oběma výstupy. Rozdíly těchto dvou způsobů jsou zanedbatelné, avšak trénink druhým způsobem je $1,5\times$ rychlejší. Jako optimalizační algoritmus použili autoři *Stochastic gradient descent*.



Obrázek 3.3: Architektura modelu CASA-Crowd. Převzato z [13].

3.4 CASA-Crowd

Jedná se o model, který vychází z práce *CASA-Crowd: A Context-Aware Scale Aggregation CNN-Based Crowd Counting Technique* [13], která se zabývá počítáním lidí v davu. S problémem rozdílné perspektivy a různými velikostmi objektů se snaží vypořádat pomocí vrstev dilatované konvoluce, podobně jako model Scale Pyramid Network.

Architektura tohoto modelu je znázorněna na obrázku 3.3 a je rozdělena na dvě části. První část s názvem *DFEN* je převzatá z předtrénované sítě VGG16 a obsahuje filtry o stejné velikosti 3×3. Tato část je schopna extrahovat jednoduché i komplexní příznaky ze vstupního obrázku. Skládá se ze čtyř bloků, kde každý z nich provádí několik operací konvoluce. Tyto bloky postupně zpracovávají data od jednoduchých příznaků až po složitější. Bloky 1-2 jsou zodpovědné za získávání příznaků menších velikostí jako jsou například tečky, čáry a křivky. Bloky 3-4 získávají z dat komplexnější příznaky jako jsou rohy a hrany.

Následuje druhá část modelu s názvem *SAD*, která je rozdělena na tři bloky. V každém z těchto bloků jsou paralelně zapojené vrstvy dilatované konvoluce s různými velikostmi filtrů a různými parametry *dilation rate*. Díky tomu je síť schopna pracovat s různou velikostí objektů. Filtry s menší velikostí se zaměřují hlavně na menší objekty, zatímco filtry s větší velikostí se zaměřují na ty větší. Výstup každého větvě je spojen dohromady a předán následujícímu bloku.

Použitím dilatované konvoluce v části SAD se výrazně zvyšuje schopnost sítě selektivně agregovat víceúrovňové kontextové informace, aniž by bylo zapotřebí při trénování využívat jakékoli perspektivní mapy.

Autoři k tréninku používají optimalizační algoritmus *Stochastic Gradient Descent* a chybovou funkci *Euclidean loss*. Síti se na vstupu předávají vzorky o velikosti 1/4 původního obrázku a výstupem je mapa hustoty o velikosti 1/8 vstupního vzorku.

3.5 Multi-scale Fusion Network

Multi-scale Fusion Network neboli SFANet vychází z práce *Dual Path Multi-Scale Fusion Networks with Attention for Crowd Counting* [29]. Tento model využívá mapu pozornosti k tomu, aby zdůraznil oblasti, kde se objekty nachází. Díky tomu se lze vypořádat s nerovnoměrným rozložením objektů ve scéně a také lze z mapy hustoty odstranit šum na pozadí.

Jedná se o multi-task model, který se skládá ze dvou částí. První část s názvem FME (*conv1-1* až *conv5-3*) je převzatá z předtrénované sítě VGG16 a stará se o extrahování příznaků ze vstupního obrázku. Druhá část je dvouvětvá síť víceúrovňové fúze příznaků, která se stará o generování mapy hustoty. První větev s názvem *Attention map path* (AMP) je zodpovědná za generování tzv. *mapy pozornosti* a druhá větev s názvem *Density map path* (DMP) vytváří mapu hustoty. Větev DMP kombinuje mapu pozornosti z větve AMP se svou mapou příznaků a tím vytvoří finální mapu hustoty.

Architekturu modelu SFANet lze vidět na obrázku 3.4. Vstupem mohou být jak vzorky obrázků, tak i celý originální snímek. FME tento vstup zpracuje a výsledné mapy příznaků posílá do větví AMP a DMP. Tyto mapy příznaků jsou v různých velikostech – 1/2, 1/4, 1/8, 1/16. Větvě AMP a DMP tyto mapy příznaků postupně spojují, čímž získávají informace z víceúrovňových příznaků a model díky tomu dokáže lépe pracovat s rozdílnou velikostí objektů. Výstupní mapy hustoty a pozornosti mají velikost 1/2 vstupního obrázku.

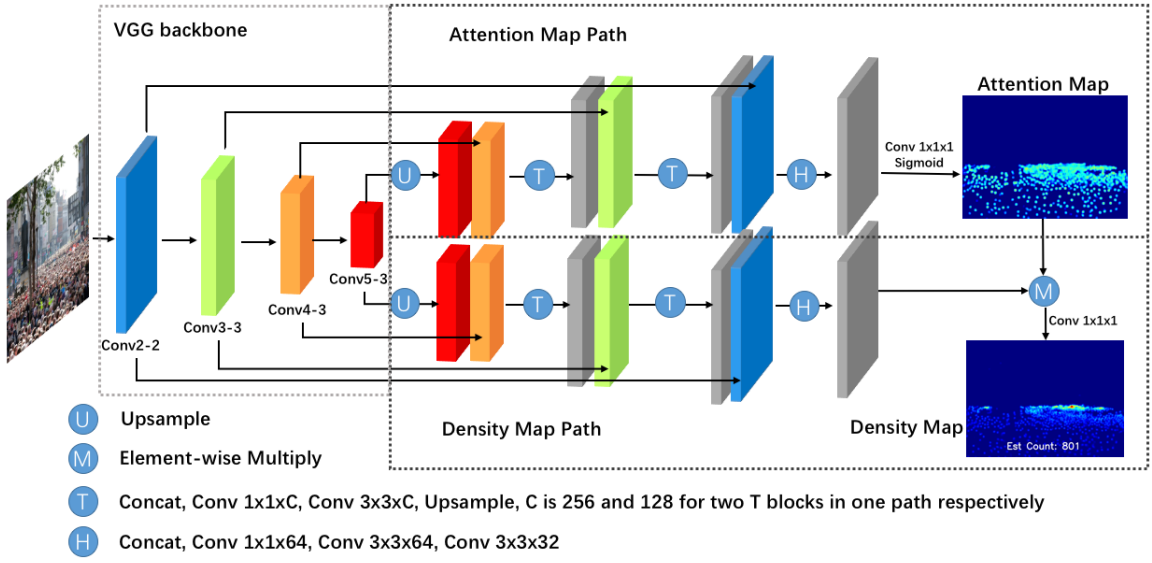
Mapa pozornosti určuje pravděpodobnost výskytu objektu pro každý pixel v mapě hustoty. V nejlepším případě je instance objektu označena 1 a vše ostatní 0. Při trénování

předáváme síti kromě ground truth mapy hustoty také ground truth mapu pozornosti. Vznik těchto map je popsán v sekci 4.2.

Autoři při trénování použili optimalizační algoritmus *Adam* a chybové funkce *Euclidean loss* pro výstup mapy hustoty a *Binary cross entropy* pro výstup mapy pozornosti, která je definována vztahem 3.3.

$$L_{att} = -\frac{1}{N}\sum_{i=1}^{N}(A_i^{GT}\ log(P_i) + (1 - A_i^{GT})\ log(1 - P_i))\,, \tag{3.3}$$

kde A_i^{GT} značí ground truth mapu pozornosti, P_i udává pravděpodobnost každého pixelu aktivovaného funkcí sigmoid v odhadované mapě pozornosti a N je počet snímků v trénovací dávce.



Obrázek 3.4: Architektura modelu SFANet. Převzato z [29].

Kapitola 4

Trénování a experimenty

V této kapitole popisuji, jak probíhala reimplementace uvedených modelů, jaké knihovny a datové sady jsem k tomu použil a také jak probíhal proces trénování.

Veškerá implementace proběhla v programovacím jazyce Python[1]. Jako knihovnu pro implementaci neuronových sítí jsem zvolil Keras[2], která funguje jako rozhraní pro knihovnu TensorFlow[3].

Trénování všech modelů probíhalo na platformě Google Colaboratory[4], která využívá prostředí Jupyter notebook[5]. Veškeré výpočty zde probíhají v cloudu a většina knihoven je již předem nainstalována. Nevýhodou Google Colaboratory je, že se nedá vybrat konkrétní grafická karta, na které výpočty poběží. Ve verzi zdarma je také omezený čas běhu na 12 hodin a není zde možnost běhu na pozadí. Tyto limity se dají změnit zakoupením verze Colab Pro, případně Colab Pro+. Tarif Colab Pro+ umožňuje přednostní přístup k virtuálním počítačům s větší pamětí a rychlejšími GPU. Zároveň je zde umožněn běh na pozadí a čas běhu je prodloužen na 24 hodin, proto jsem využil právě této možnosti.

K ukončení trénování jsem použil *EarlyStopping callback* z knihovny Keras, který dokáže odhalit začátek přeučení a následně ukončit trénování modelu.

Pro zpracování obrázků jsem použil knihovny OpenCV[6], SciPy[7] a scikit-image[8]. Datové sady jsem virtuálním počítačům zpřístupnil pomocí Google Drive.

4.1 Datové sady

K trénování a testování modelů byly použity tři datové sady. Jedná se o datové sady TRANCOS[9], CARPK[10] a PUCPR+[10]. Každá z těchto datových sad obsahuje snímky s různou perspektivou a různými světelnými podmínkami.

[1] https://www.python.org/
[2] https://keras.io/
[3] https://www.tensorflow.org/
[4] https://research.google.com/colaboratory
[5] https://jupyter.org/
[6] https://pypi.org/project/opencv-python/
[7] https://scipy.org/
[8] https://scikit-image.org/
[9] https://gram.web.uah.es/data/datasets/trancos/index.html
[10] https://lafi.github.io/LPN/

Obrázek 4.1: Ukázka datové sady TRANCOS

TRANCOS

TRaffic ANd COngestionS neboli TRANCOS je datová sada, která byla publikována v práci *Extremely Overlapping Vehicle Counting* [8] a obsahuje snímky pořízené z dopravních kamer s různým osvětlením, perspektivou a hustotou dopravy. Datová sada je tvořena 1 244 snímky obsahujících 46 796 anotovaných vozidel a je rozdělena na tři části – trénovací (403 snímků), validační (420 snímků) a testovací (421 snímků). Tyto části musí být využity pouze na svůj daný účel. Každý snímek má rozměry 640×480 pixelů.

Vozidla jsou anotována pomocí souřadnic jejich středů. Ne všechny vozidla na obrázků jsou však anotována a proto je ke každému snímku přiložen matlab soubor s jeho binární maskou, která určuje oblast zájmu (region of interest).

Ukázka datové sady TRANCOS je zobrazena na obrázku 4.1.

CARPK

The Car Parking Lot Dataset (CARPK) byl publikován v práci *Drone-based Object Counting by Spatially Regularized Regional Proposal Network* [12].

Tato datová sada obsahuje 1 448 snímků s celkem 89 777 anotovanými vozidly. Záběry byly pořízeny ze čtyř různých parkovišť pomocí dronu a proto obsahuje pouze jednu perspektivu. Každý snímek má rozměry 1280×720 pixelů.

CARPK je rozdělen do dvou částí – trénovací (989 snímků) a testovací (459 snímků). Validační část není určena, proto jsem z trénovací části odebral 20 % snímků, které jsou využity k validaci.

Ukázku datové sady CARPK lze vidět na obrázku 4.2.

Obrázek 4.2: Ukázka datové sady CARPK

Obrázek 4.3: Ukázka datové sady PUCPR+

PUCPR+

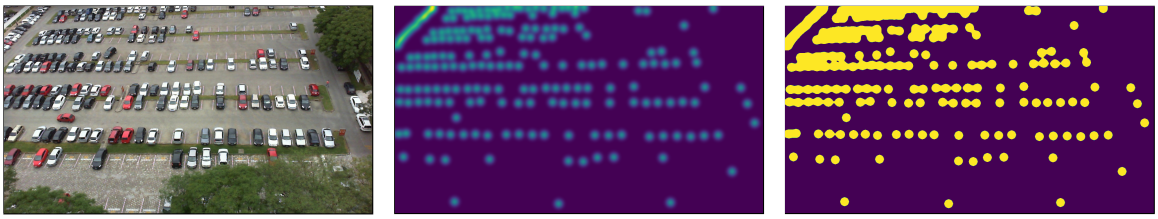
Pontifical Catholic University of Parana+ Dataset (PUCPR+) obsahuje fotografie s různým počasím pořízené z desátého patra budovy a proto mají pouze jednu perspektivu. Tato datová sada byla publikována společně s CARPK v práci *Drone-based Object Counting by Spatially Regularized Regional Proposal Network* [12].

Obsahuje 125 snímků s celkem 16 456 anotovanými vozidly a je rozdělena na dvě části – trénovací (100 snímků) a testovací (25 snímků). Podobně jako CARPK neobsahuje validační část, proto jsem z trénovací části odebral 20 % snímků a použil je k validaci.

Ukázku datové sady PUCPR+ lze vidět na obrázku 4.3.

4.2 Příprava groud truth dat

K trénování každého modelu je zapotřebí pro každý snímek vytvořit ground truth mapu hustoty a v případě modelu SFANet také ground truth mapu pozornosti.



Obrázek 4.4: Ukázka mapy hustoty a pozornosti na snímku z PUCPR+

Mapa hustoty

Pro vznik ground truth map hustot je zapotřebí mít vhodně anotovaná vozidla. Datová sada TRANCOS používá bodové anotace středů vozidel, zatímco CARPK a PUCPR+ používají anotace pomocí ohraničených rámečků (bounding box). Ze souřadnic rámečků lze však snadno určit jejich střed a proto není žádný problém je převést na bodové anotace.

Ground truth mapu hustoty lze vytvořit aplikací Gaussovy funkce na souřadnice anotovaných objektů. Výsledná mapa hustoty je dána vztahem 4.1.

$$D_I(p) = \sum_{\mu \in A_I} \mathcal{N}(p; \mu, \Sigma)\,, \tag{4.1}$$

kde A_I je sada anotovaných bodů pro obrázek I. $\mathcal{N}(p; \mu, \Sigma)$ znázorňuje aplikování normalizované 2D Gaussovy funkce se střední hodnotou μ a izotropní kovarianční maticí Σ na pixel p [18]. Pomocí mapy hustoty D_I můžeme poté spočítat celkový počet objektů N_I, který lze získat sečtením všech hodnot mapy hustoty. Tento vztah je definován rovnicí 4.2.

$$N_I(p) = \sum_{p \in I} D_I(p)\,. \tag{4.2}$$

Po sečtení je zachován původní počet objektů, i když se objekty překrývají.

Jelikož konvoluční neuronové sítě často využívají pooling vrstvy, tak výsledná mapa hustoty bývá menší než vstupní obrázek. Mapu hustoty tedy musíme upravit na požadovanou velikost. Při zvětšení či zmenšení této mapy však není zachován skutečný počet objektů, proto je potřeba při každé změně velikosti mapu hustoty normalizovat. Tato operace je dána vztahem 4.3.

$$\hat{D}^P_{pred} = \frac{\sum_{\forall p} D^P_{pred}(p)}{\sum_{\forall p} \hat{D}^P_{pred}(p)} \hat{D}^P_{pred}\,, \tag{4.3}$$

kde D^P_{pred} udává mapu hustoty s původní velikostí, $\hat{D}^P_{pred}$ určuje mapu hustoty se změněnou velikostí a p je pixel z mapy hustoty [18].

Mapa pozornosti

Ground truth mapy pozornosti využívá model SFANet. Tyto mapy vznikají z map hustot, jejich vznik je definován vztahem 4.4.

$$\forall x \in \mathbb{Z}, A^{GT}_i(x) = \begin{cases} 0 & x < th \\ 1 & x \geq th \end{cases}\,, \tag{4.4}$$

kde th udává práh pozornosti, $\mathbb{Z}$ je mapa hustoty a A^{GT}_i je výsledná mapa pozornosti [29]. V této práci bylo th při experimentech nastaveno na hodnotu 5e−4.

4.3 Scale Pyramid Network

Při trénování jsem postupoval podobným způsobem jako autoři této sítě. Z každého obrázku bylo vyextrahováno 9 vzorků o velikosti 224×224 pixelů. Každý vzorek byl poté horizontálně převrácen. Z jednoho snímku tedy vzniklo 18 vzorků.

Ground truth mapa hustoty byla vygenerována z původního obrázku s parametrem $\Sigma = 15$. Z této mapy hustoty byly vyextrahovány vzorky stejným způsobem jako z původního obrázku. Každý ze vzorků ground truth mapy hustoty byl poté zmenšen na velikost výstupu sítě a následně normalizován pro zachování počtu vozidel.

Pro trénování modelu jsem zvolil optimalizační algoritmus *Adam* s parametrem learning rate 1e−5. Jako chybová funkce byla zvolena *Mean squared error*, která je definována v sekci 2.3. Velikost dávky (batch size) byla nastavena na hodnotu 10.

Model nebyl trénován od základů, jelikož jeho část využívá předtrénovanou síť VGG16. Trénování proběhlo na třech zmíněných datových sadách. Na datové sadě TRANCOS narazil na začátek přeučení po 153. epoše. Na CARPK bylo trénování ukončeno po 154. epoše a na PUCPR+ po 116. epoše.

Testování proběhlo metodou posuvného okénka s posuvem 10 pixelů. Z každého extrahovaného vzorku byla sítí vygenerována mapa hustoty. Tyto mapy hustoty byly poté zvětšeny do původní velikosti, normalizovány a spojeny do finální mapy hustoty. Tato finální mapa hustoty byla následně vydělena mapou počtu překrytí.

4.4 Scale-adaptive CNN

Způsob trénování Scale-adaptive CNN proběhl podobně jako u modelu Scale Pyramid Network. Z každého obrázku bylo vyextrahováno 9 vzorků o velikosti 224 × 224 pixelů a k nim byly vytvořeny odpovídající mapy hustoty s parametrem $\Sigma = 15$. Veškeré vzorky byly poté horizontálně převráceny, aby se zvětšilo množství trénovacích dat.

K trénování byl zvolen optimalizační algoritmus *Adam* s hodnotou learning rate nastavenou na 1e−7 pro datovou sadu TRANCOS, 1e−6 pro datovou sadu CARPKP a $1,5e{-}6$ pro datovou sadu PUCPR+. Jako chybová funkce byla zvolena *Mean squared error* pro výstup mapy hustoty a *Relative count loss* pro počet vozidel. Koeficienty chybových funkcí byly nastaveny následovně – 1,0 pro Mean squared error a 0,1 pro Relative count loss. Síť byla trénována s oběma výstupy zároveň jako multi-task model.

Model u datové sady TRANCOS narazil na začátek přeučení po 305. epoše. U datové sady CARPK začátek přeučení nastal po 605. epoše a u PUCPR+ po 444. epoše.

Testování proběhlo stejným způsobem jako u modelu Scale Pyramid Network. Bylo použito posuvné okénko s posuvem 10 pixelů, které postupně extrahovalo části obrázku a předávalo je síti, která vygenerovala jejich mapy hustoty. Tyto mapy hustoty byly poté spojeny do jedné finální mapy hustoty, která byla vydělena mapou počtu překrytí.

4.5 CASA-Crowd

Generování dat pro tento model proběhlo stejným způsobem jako pro předešlé modely. Z trénovacích obrázků bylo vyextrahováno 9 vzorků o velikosti 224×224 pixelů, které byly poté horizontálně převráceny pro větší množství trénovacích dat. K těmto vzorkům se následně vygenerovaly mapy hustot s parametrem $\Sigma = 15$.

CASA-Crowd nebyl trénován od základů, protože používá část předtrénované sítě VGG16. Jako optimalizační algoritmus byl zvolen *Adam* s parametrem learning rate 1e−7 pro datovou sadu TRANCOS a 1e−6 pro datové sady CARPK a PUCPR+. Jako chybovou funkci jsem použil *Mean squared error*. Model narazil u datové sady TRANCOS na začátek přeučení po 308. epoše. Na datové sadě CARPK nastal začátek přeučení po 268. epoše a na PUCPR+ po 152. epoše.

Testování modelu proběhlo stejným způsobem jako u modelu Scale Pyramid Network, kde je použité posuvné okénko s posuvem 10 pixelů pro extrakci jednotlivých vzorků. Z každého vzorku byla poté vygenerována mapa hustoty a následně byly tyto mapy zvětšeny, normalizovány a spojeny do finální mapy hustoty. Tato finální mapa hustoty se poté vydělila mapou počtu překrytí.

4.6 Multi-scale Fusion Network

Data pro Multi-scale Fusion Network (SFANet) byla generována podobným způsobem jako u autorů tohoto modelu. Z každého trénovacího obrázku bylo extrahováno 25 vzorků o velikosti 400×400 pixelů. Tyto vzorky měly pravděpodobnost 0,5 pro horizontální převrácení a 0,3 pro změnu gamma kontrastu v rozmezí od 0,5 do 1,5. Pro každý vzorek byla vygenerována mapa hustoty s parametrem $\Sigma = 10$.

Trénování modelu neprobíhalo od základů, jelikož tento model využívá část předtrénované sítě VGG16. Jako optimalizační algoritmus jsem zvolil *Adam* s parametrem learning rate 2e−5 pro datovou sadu TRANCOS a 5e−5 pro datové sady CARPK a PUCPR+. Byly použity dvě chybové funkce – *Mean squared error* (MSE) pro výstup mapy hustoty a *Binary cross entropy* (BCE) pro výstup mapy pozornosti. Koeficienty těchto funkcí měly hodnoty 1,0 pro MSE a 0,1 pro BCE.

Model narazil na začátek přeučení po 123. epoše u datové sady TRANCOS. U datové sady CARPK započalo přeučení po 143. epoše a u PUCPR+ po 370. epoše.

Vyhodnocování proběhlo podobně jako u předešlých modelů. Použilo se posuvné okénko s posuvem 20 pixelů. Pro každý extrahovaný vzorek byla sítí vygenerována mapa hustoty. Tyto mapy hustoty byly poté spojeny do finální mapy hustoty a tato finální mapa hustoty byla následně vydělena mapou počtu překrytí.

Kapitola 5

Dosažené výsledky

V této kapitole popisuji metriky, které byly použity při vyhodnocování zmíněných modelů. Vyhodnocování proběhlo na stejných datových sadách, na kterých byly natrénovány. Proběhlo také křížové vyhodnocení pro otestování schopnosti generalizace, kdy byly modely vyhodnoceny na jiných datových sadách, než na kterých byly natrénovány. Dále jsou modely porovnány podle jejich rychlosti a složitosti. V neposlední řadě jsou také porovnány výsledné mapy hustot všech modelů.

5.1 Metriky

Jednou z nejpoužívanějších metrik pro vyhodnocení odhadu hustoty je *Mean absolute error* (MAE). Tato metrika počítá absolutní rozdíl mezi odhadovaným a očekávaným počtem. Její rovnice je dána vztahem 5.1.

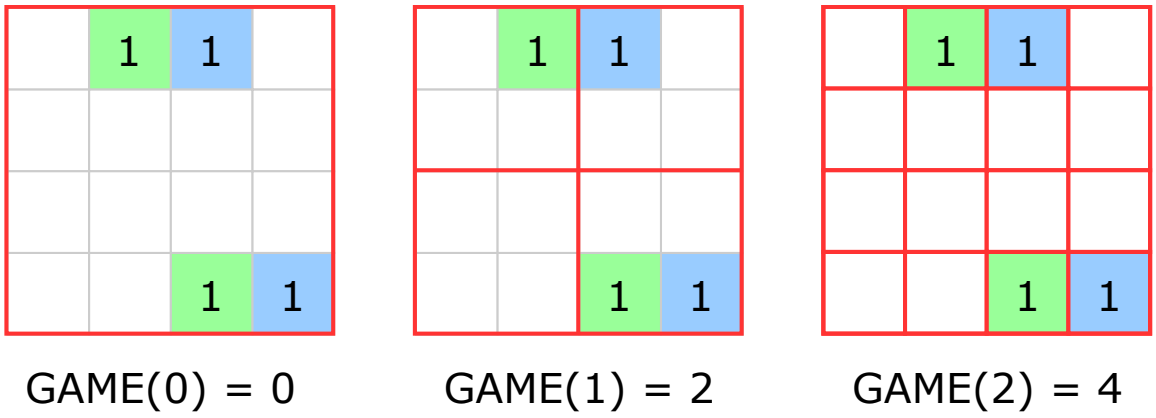
$$MAE = \frac{1}{N}\sum_{i=1}^{N}|y_i - \hat{y}_i|\,, \tag{5.1}$$

kde N udává celkový počet testovacích snímků, y_i je ground truth počet objektů pro snímek i a $\hat{y}_i$ je modelem odhadovaný počet objektů pro stejný snímek i.

Další metrikou je *Grid Average Mean absolute Error* (GAME), která pochází z práce *Extremely Overlapping Vehicle Counting* [8]. Tato metrika bere v potaz lokalizační informaci počítaných objektů a je dána vztahem 5.2.

$$GAME(L) = \frac{1}{N}\sum_{n=1}^{N}(\sum_{l=1}^{4^L}|e_n^l - gt_n^l|)\,, \tag{5.2}$$

kde N udává celkový počet testovacích snímků, e_n^l je odhadovaný počet objektů snímku n v oblasti l, gt_n^l je ground truth počet objektů pro stejný snímek ve stejné oblasti. L udává úroveň metriky. Čím větší je L, tím přesnější metrika je. GAME(0) odpovídá metrice MAE. Grafické znázornění metriky GAME je ukázáno na obrázku 5.1.



Obrázek 5.1: Princip metriky GAME. Zelená oblast označuje ground truth objekty ve snímku a jejich počty. Modrá oblast označuje odhadované pozice objektů a jejich počty. Na obrázku lze vidět, že výsledek GAME(0) neboli MAE je roven 0, jelikož nebere v potaz lokalizační informaci. Při vyšších hodnotách L se chybná lokalizace více penalizuje.

5.2 Výsledky datové sady TRANCOS

Tabulka 5.1 uvádí výsledky modelů trénovaných a vyhodnocených na datové sadě TRANCOS.

Model	GAME(0) = MAE	GAME(1)	GAME(2)	GAME(3)
SPN [3]	3,35	4,94	6,47	9,22
SPN	5,44	**6,74**	**8,22**	**9,95**
CASA-Crowd	**5,14**	7,49	10,00	12,65
SaCNN	6,29	8,50	11,41	14,42
SFANet	9,64	10,48	11,76	13,60

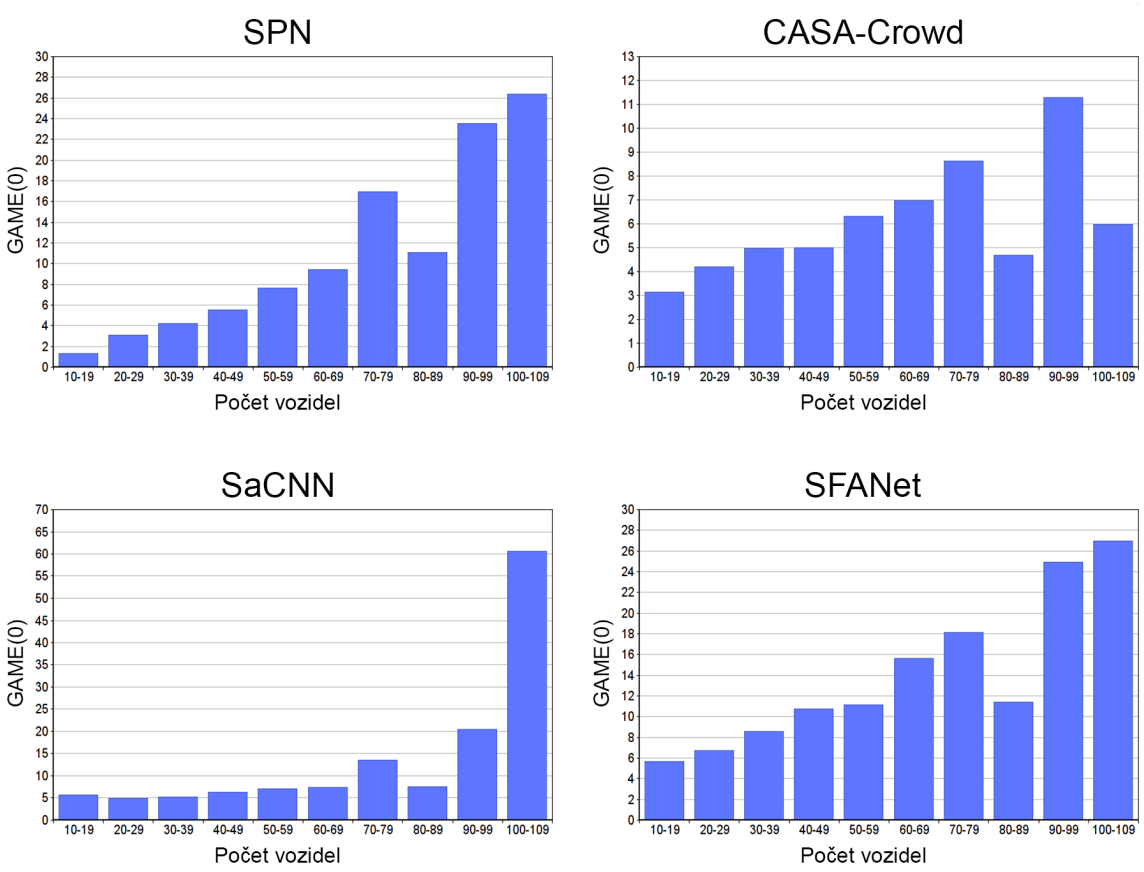
Tabulka 5.1: Výsledky modelů trénovaných a vyhodnocených na datové sadě TRANCOS. Tučně označené hodnoty uvádí nejlepší výsledek mnou trénovaných modelů.

Autoři modelu Scale Pyramid Network (SPN) provedli testování na datové sadě TRANCOS, proto jsem jejich výsledky uvedl v prvním řádku tabulky za účelem porovnání. Při porovnání mých výsledků s původními lze vidět, že se autorům povedlo dosáhnout lepších výsledků. Rozdíl při GAME(0) je 2,09, avšak při GAME(3) je rozdíl pouze 0,73. U mnou trénovaného modelu je tedy odhad počtu vozidel horší, ale schopnost lokalizace je srovnatelná.

Na modelu CASA-Crowd vyšla GAME(0) o 0,3 méně než u mnou trénovaného SPN, avšak všechny ostatní úrovně dosahují horších výsledků. Schopnost odhadu počtu vozidel je tedy srovnatelná s modelem SPN, ale lokalizace vozidel zaostává. Scale-adaptive CNN (SaCNN) dosahuje ve všech úrovních horších výsledků než předchozí dva modely, ale v porovnání s modelem CASA-Crowd nejsou tyto rozdíly příliš veliké. Nejvíce se liší v GAME(3) o hodnotu 1,77. SFANet na datové sadě TRANCOS dosáhl nejhorších výsledků. V porovnání s modelem SaCNN má o 3,35 větší GAME(0), ale na vyšších úrovních se hodnotami pomalu přibližuje. Hodnotu GAME(3) má o 0,82 nižší něž model SaCNN. Schopnost správné lokalizace má tedy o trochu lepší než model SaCNN, ale v odhadování počtu je značně horší.

Na obrázku 5.2 lze vidět distribuci metriky GAME(0) pro datovou sadu TRANCOS v závislosti na ground truth počtu vozidel. Na všech grafech lze vidět, že s rostoucím počtem vozidel metrika GAME(0) stoupá. To je způsobeno tím, že s rostoucím počtem vozidel roste také možnost udělat chybu v predikci. Datová sada TRANCOS také obsahuje vozidla, která se velmi často překrývají, což znesnadňuje jejich správnou predikci.

Grafy modelů SPN a SFANet jsou si velmi podobné. S rostoucím počtem vozidel u nich stabilně stoupá metrika GAME(0). Výjimkou je hodnota pro počet vozidel 80 - 89, jelikož v této kategorii se nachází mnohem méně obrázků než v ostatních. Model SaCNN má ze všech modelů nejhorší predikci v kategorii 100 - 109. Při tak velkém počtu vozidel tedy není schopen vozidla správně počítat. CASA-Crowd má naopak ze všech modelů nejlepší predikci v kategorie 100 - 109, kde jeho hodnota GAME(0) dosahuje pouze 5,97.



Obrázek 5.2: Distribuce GAME(0) v závislosti na počtu vozidel pro datovou sadu TRANCOS

5.3 Výsledky datové sady CARPK

Tabulka 5.2 uvádí výsledky modelů trénovaných a vyhodnocených na datové sadě CARPK.

Pro porovnání výsledků jsem do tabulky přidal výsledky modelu Multi-Column CNN, který byl reimplementován a natrénován na datové sadě CARPK autory práce *Density-Based Vehicle Counting with Unsupervised Scale Selection* [6].

Model	GAME(0) = MAE	GAME(1)	GAME(2)	GAME(3)
MCNN [6]	18,16	23,74	27,32	30,39
SPN	21,53	**21,73**	**22,47**	**24,20**
CASA-Crowd	25,78	25,97	26,73	28,74
SaCNN	**21,10**	24,09	30,82	38,74
SFANet	21,87	22,57	23,69	26,17

Tabulka 5.2: Výsledky modelů trénovaných a vyhodnocených na datové sadě CARPK. Tučně označené hodnoty uvádí nejlepší výsledek mnou trénovaných modelů.

Model SPN v metrice GAME(0) nepřekonal výsledky modelu ze zmíněné práce, avšak ve vyšších úrovních této metriky dosáhl lepších výsledků. SPN má tedy horší odhad počtu vozidel, ale lokalizovat je dokáže s větší přesností. CASA-Crowd dosáhl ve všech úrovních horších výsledků než model SPN. Tento rozdíl je v rozmezí od 4,24 do 4,54. Od úrovně GAME(2) však překonal model MCNN. Scale-adaptive CNN v metrice GAME(0) dosáhl nejlepšího výsledku ze všech čtyř reimplementovaných modelů, avšak model MCNN nepřekonal. V metrikách GAME(2) a GAME(3) dosáhl nejhorších výsledků z celé datové sady. V porovnání s ostatními modely je tedy v lokalizaci vozidel nejhorší. Model SFANet dosahuje průměrných výsledků. V metrice GAME(3) je na druhém místě s rozdílem 1,97 oproti SPN. Díky tomu je model SFANet druhým nejpřesnějším modelem z hlediska lokalizace vozidel.

Na obrázku 5.3 lze vidět distribuci metriky GAME(0) pro datovou sadu CARPK v závislosti na ground truth počtu vozidel. Grafy modelů SPN, CASA-Crowd a SFANet jsou si velmi podobné. Podobně jako na grafech z datové sady TRANCOS lze vidět, že s rostoucím počtem vozidel metrika GAME(0) stoupá. Výjimkou je model SaCNN, který má značně zhoršenou predikci vozidel v kategorii 0 - 19. Tento model dosahuje také nejhorší predikce v kategorii vozidel 180 - 199, kdy metrika GAME(0) dosahuje hodnot 63,90

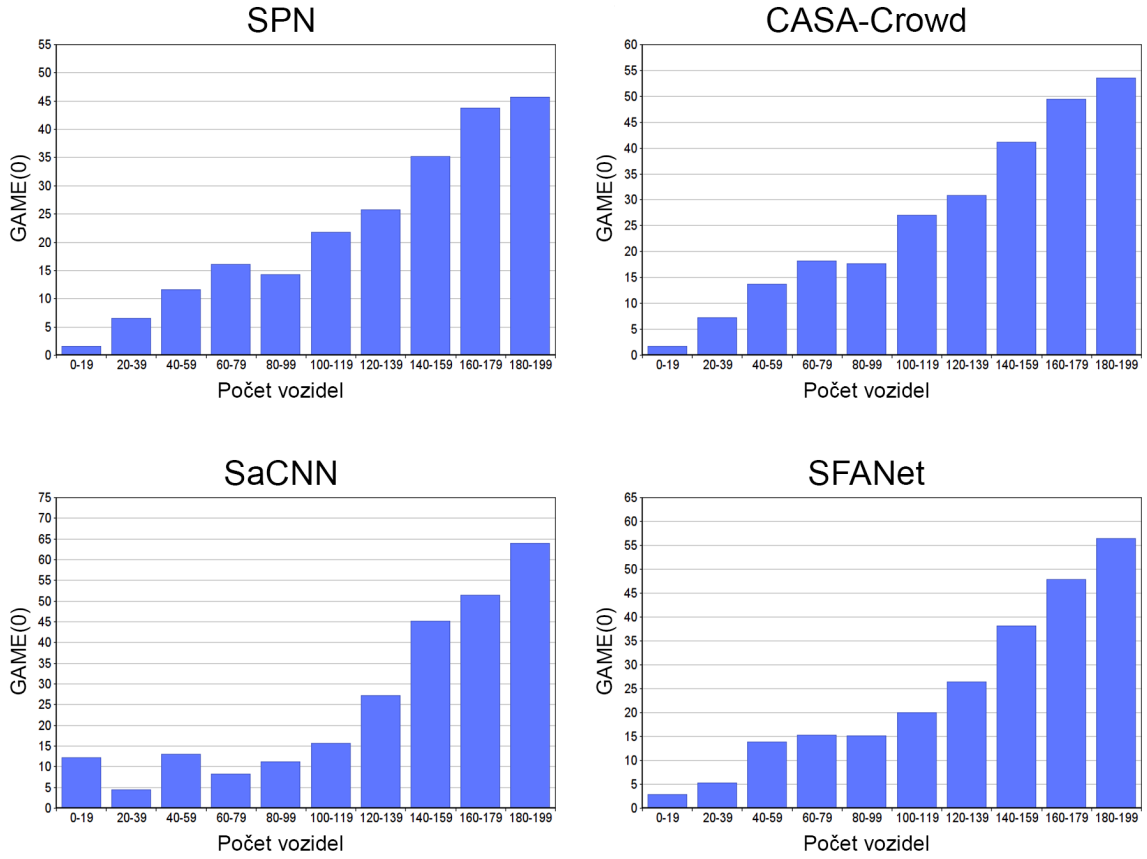
5.4 Výsledky datové sady PUCPR+

Tabulka 5.3 uvádí výsledky modelů trénovaných a vyhodnocených na datové sadě PUCPR+.

Model	GAME(0) = MAE	GAME(1)	GAME(2)	GAME(3)
MCNN [6]	26,49	28,82	29,95	30,95
SPN	9,01	**9,17**	**11,27**	**13,89**
CASA-Crowd	9,19	11,23	14,67	19,36
SaCNN	28,01	42,36	46,36	54,06
SFANet	**8,35**	11,79	13,87	18,28

Tabulka 5.3: Výsledky modelů trénovaných a vyhodnocených na datové sadě PUCPR+. Tučně označené hodnoty uvádí nejlepší výsledek mnou trénovaných modelů.

Podobně jako v tabulce 5.2 jsem i do této tabulky pro porovnání přidal výsledky modelu MCNN z práce *Density-Based Vehicle Counting with Unsupervised Scale Selection* [6], jelikož byl tento model reimplementován a natrénován na datové sadě PUCPR+.



Obrázek 5.3: Distribuce GAME(0) v závislosti na počtu vozidel pro datovou sadu CARPK

Model SPN dosáhl velmi dobrých výsledků v porovnání s ostatními modely. Od metriky GAME(1) se jedná o nejlepší model z celé datové sady. CASA-Crowd se v metrice GAME(0) vyrovnal modelu SPN s rozdílem pouhých 0,18. Správný počet vozidel tedy dokáže určit stejně jako model SPN, ale od metriky GAME(1) se postupně zhoršuje. SaCNN dosáhl ve všech úrovních GAME metriky nejhorších výsledků ze všech modelů. V metrice GAME(0) se přiblížil modelu MCNN s rozdílem 1,52, ale ve všech ostatních úrovních této metriky se výrazně zhoršil. Model SFANet dosáhl nejlepšího výsledku v metrice GAME(0) s rozdílem pouhých 0,66 oproti modelu SPN. V ostatních úrovních této metriky je však horší. V metrice GAME(3) dosáhl druhého nejlepšího výsledku s rozdílem 4,39 oproti SPN.

Distribuce metriky GAME(0) v závislosti na počtu vozidel nebyla zjištěna, jelikož datová sada PUCPR+ obsahuje pouze 25 testovacích snímků.

5.5 Výsledky křížového vyhodnocení

Tabulka 5.4 uvádí výsledky křížového vyhodnocení modelů trénovaných na datových sadách TRANCOS a CARPK a testovaných na datové sadě PUCPR+.

Z tabulky můžeme vidět, že u všech modelů narostly hodnoty metrik. Je to způsobené převážně tím, že se testovací obrázky velmi liší od těch, co byly v trénovacích sadách. Nejlepších výsledků ve všech metrikách dosáhl model SFANet, který byl trénován na datové sadě TRANCOS. Druhým nejlepším modelem je SaCNN trénovaný na datové sadě TRANCOS,

Model	GAME(0) = MAE	GAME(1)	GAME(2)	GAME(3)
SPN (T)	62,13	100,85	117,09	122,67
SPN (C)	110,47	110,57	110,89	111,15
CASA-Crowd (T)	112,64	176,43	191,92	199,96
CASA-Crowd (C)	119,13	119,25	119,43	119,59
SaCNN (T)	58,72	67,34	77,05	83,93
SaCNN (C)	104,09	105,47	106,10	106,79
SFANet (T)	**18,41**	**28,19**	**37,01**	**45,80**
SFANet (C)	132,34	132,95	133,92	135,00

Tabulka 5.4: Výsledky křížového vyhodnocení modelů trénovaných na datových sadách TRANCOS (T) a CARPK (C) a testovaných na datové sadě PUCPR+.

který má GAME(0) o 40,31 větší než model SFANet, který byl trénován na datové sadě TRANCOS. V metrice GAME(3) se liší o 38,13. SPN trénovaný na datové sadě TRANCOS je třetím nejlepším modelem z hlediska odhadu správného počtu vozidel. V GAME(0) se od SaCNN (T) liší pouze o 3,41. Naopak ve správné lokalizaci vozidel ho překonal model SaCNN trénovaný na datové sadě CARPK. Tento model má o 15,88 nižší GAME(3).

Modely trénované na datové sadě TRANCOS dosahovaly ve většině případů lepších výsledků. Je to z toho důvodu, že se snímky z datové sady TRANCOS více podobají snímkům z PUCPR+ z hlediska perspektivy.

5.6 Porovnání vlastností modelů

V této sekci se zabývám porovnáním rychlostí a složitostí daných modelů. Také zde porovnám jejich výsledné mapy hustoty.

Porovnání rychlosti

Tabulka 5.5 uvádí rychlosti vyhodnocení jednotlivých modelů. Testované obrázky měly rozměry 640×480 pixelů. Vyhodnocení proběhlo pro jednotlivé vzorky i pro celé obrázky.

Testování rychlosti vzorků proběhlo následujícím způsobem. Z dvaceti testovacích obrázků bylo vyextrahováno dvacet vzorků o velikosti 400×400 pixelů pro model SFANet a 224×224 pixelů pro ostatní modely. K těmto vzorkům byly následně modelem predikovány jejich mapy hustoty. Časy všech vzorků se poté zprůměrovaly.

Při testování rychlosti celého snímku se z dvaceti testovacích snímků postupně extrahovaly vzorky s posuvem 20 pixelů pro model SFANet a 10 pixelů pro ostatní modely. K těmto vzorkům se následně vygenerovaly predikce jejich map hustot a z nich byla složena finální mapa hustoty. Tento čas se poté zprůměroval.

Výsledky je třeba brát pouze orientačně, jelikož se mohou postupně měnit například s počtem dodaných dat a vytížeností GPU. Model SFANet navíc používá jinou velikost vzorků a také jiný posun.

Výsledky modelu SPN vyšly jako nejhorší ze všech modelů. Za jednu sekundu dokáže predikovat pouze 27 vzorků. CASA-Crowd je nejrychlejší model z hlediska predikce vzorku. Vzorek dokáže predikovat o 16,269 milisekund rychleji než model SPN a za jednu vteřinu dokáže predikovat až 48 vzorků. SaCNN dosáhl průměrné rychlosti vzorků i celého snímku. Dokáže predikovat až 44 vzorků za vteřinu a je o 1,738 milisekund pomalejší než model

Model	Vzorek (ms)	Celý snímek (ms)
SPN	36,947	45 290,378
CASA-Crowd	**20,678**	23 123,721
SaCNN	22,416	21 745,759
SFANet	29,168	**13 494,576**

Tabulka 5.5: Rychlost vyhodnocení jednotlivých modelů. Tučně označené hodnoty uvádí nejlepší výsledek.

CASA-Crowd. SFANet dosáhl nejrychlejšího času pro celý snímek, ale rychlost predikce vzorků je druhá nejhorší ze všech modelů (34 vzorků za vteřinu). Je to způsobeno převážně tím, že používá jinou velikost vzorků a také jiný posun. Jeden vzorek dokáže predikovat o 8,49 pomaleji než CASA-Crowd.

Porovnání složitosti

Tabulka 5.6 udává celkový počet parametrů jednotlivých modelů. Počet parametrů udává složitost daného modelu. Modely s větším počtem parametrů bývají většinou přesnější a méně náročnější na vstupní data, ale jsou náročnější na výpočet.

Model	Celkový počet parametrů
SPN	34 771 265
CASA-Crowd	22 161 601
SaCNN	25 071 425
SFANet	17 014 214

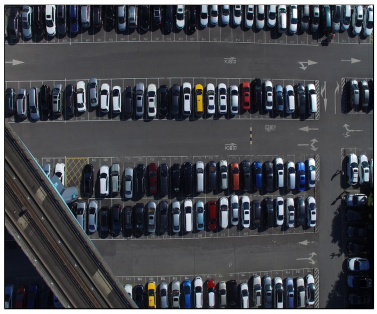
Tabulka 5.6: Celkové počty parametrů jednotlivých modelů.

SFANet obsahuje nejmenší počet parametrů a je tudíž nejjednodušším modelem ze všech testovaných. Přesto několikrát překonal výsledky ostatních modelů. Oproti němu model SPN obsahuje největší počet parametrů. SPN se ukázal jako velmi účinný z hlediska jeho dosažených výsledků, ale přesto byl v některých ohledech překonán modely s menším počtem parametrů.

Porovnání map hustot

Zde uvádím porovnání vygenerovaných map hustot podle jednotlivých modelů. Mapy hustot jsou rozděleny do sloupců podle datové sady, do které patří.

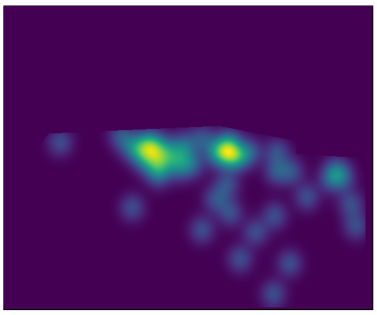
Snímek z TRANCOS



Snímek z CARPK

Snímek z PUCPR+

Obrázek 5.4: Vstupní snímek

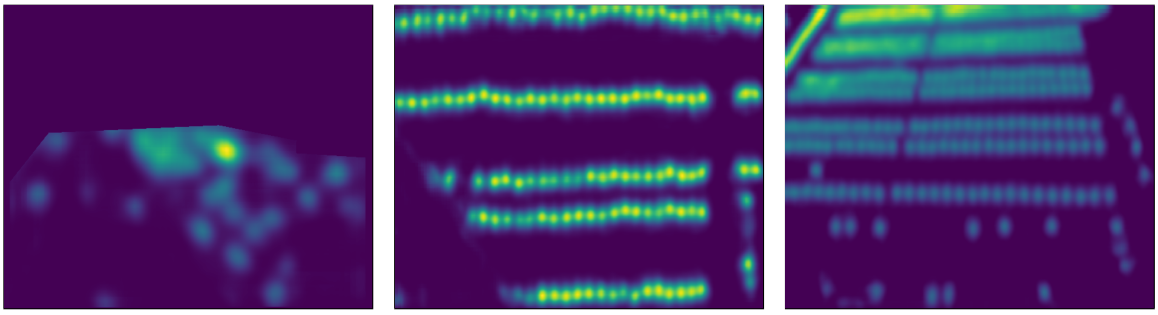


45

139

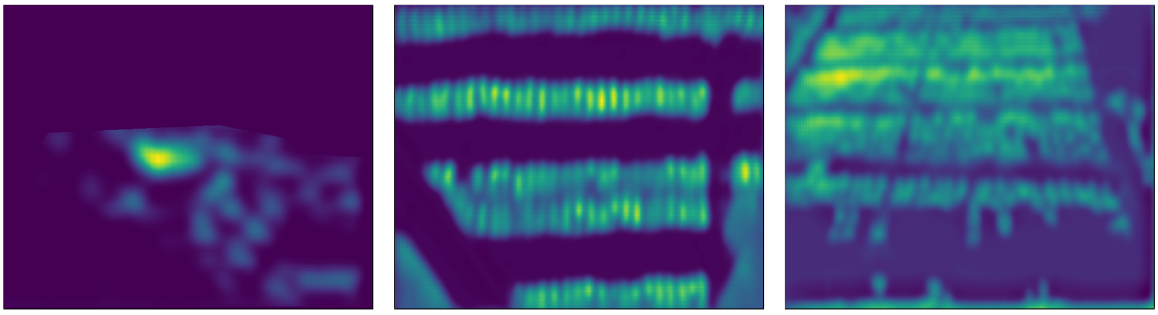
293

Obrázek 5.5: Ground truth mapy hustoty



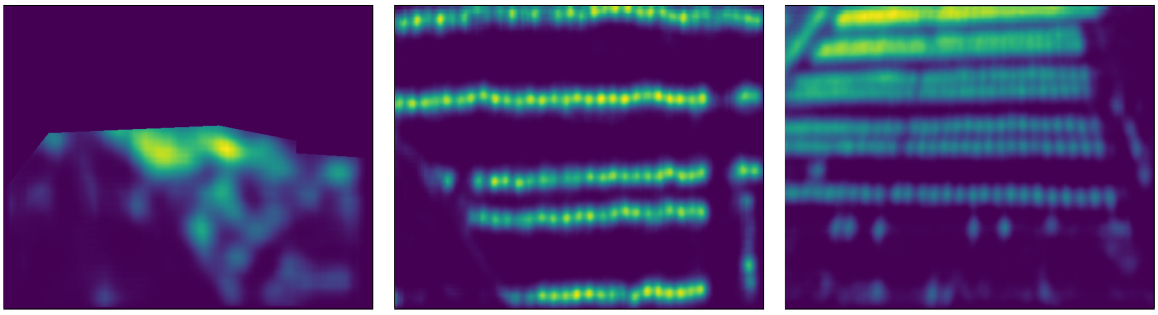
28,11 113,27 280,26

Obrázek 5.6: Odhad map hustot pomocí modelu SPN



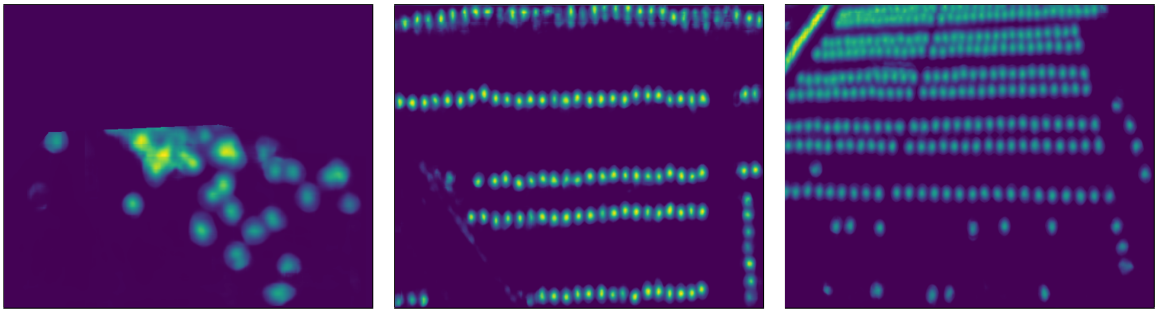
38,61 102,62 246,85

Obrázek 5.7: Odhad map hustot pomocí modelu SaCNN



38,12 112,65 276,37

Obrázek 5.8: Odhad map hustot pomocí modelu CASA-Crowd



27,47 114,38 295,54

Obrázek 5.9: Odhad map hustot pomocí modelu SFANet

Kapitola 6

Závěr

Hlavním cílem této bakalářské práce byla reimplementace a vyhodnocení modelů pro počítání vozidel ve statickém obraze se zaměřením na rozdílné velikosti vozidel ve scéně. Tyto modely fungují na principu počítání pomocí odhadu hustoty. Byly vybrány celkem čtyři modely konvolučních neuronových sítí, které byly následně reimplementovány a vyhodnoceny na datových sadách TRANCOS, CARPK a PUCPR+. Většina těchto modelů byla určena na počítání lidí v davu, ale dají se použít i na počítání vozidel.

Prvním vybraným modelem byl Scale Pyramid Network. Tento model dosáhl nejlepších výsledků na všech datových sadách. Jeho nevýhodou je, že jeho predikce snímku je nejpomalejší ze všech vybraných modelů. Predikce jednoho snímku (640×480 pixelů) trvala přibližně 45,29 sekund. Další nevýhodou je jeho složitost, jelikož obsahuje více než 34 miliónů parametrů, což je nejvíce ze všech testovaných modelů.

Dalším testovaným model byl Scale-adaptive CNN, který jako jediný z modelů neobsahuje předtrénovanou síť VGG16. Uspokojivých výsledků dosáhl pouze na datové sadě TRANCOS a CARPK. Při vyhodnocování na datové sadě PUCPR+ dosáhl výrazně horších výsledků v porovnání s ostatními modely.

Dále byl vybrán model CASA-Crowd, který dosahoval průměrných výsledků na všech datových sadách. Nejlepších výsledků dosáhl na datové sadě TRANCOS. Jeho výhodou je rychlost predikce jednotlivých vzorků. Jeden vzorek dokázal predikovat za 20,678 milisekund.

Posledním vybraným modelem byl Multi-scale Fusion Network. Tento model dosahoval velmi dobrých výsledků na všech datových sadách. Zároveň se ukázal jako nejvíce účinný v oblasti generalizace, jelikož dosáhl nejlepších výsledků v křížovém vyhodnocení. Další výhodou je jeho rychlost. Jeden snímek dokáže predikovat přibližně za 13,5 vteřiny.

Model Scale Pyramid Network dosáhl nejlepších výsledků a dokázal tak, že extrakce příznaků různých velikostí za pomoci několika paralelních vrstev dilatované konvoluce je velmi úspěšná. V budoucnu bych se zaměřil právě na modely využívající dilatovanou konvoluci.

Literatura

[1] Albawi, S., Mohammed, T. A. a Al Zawi, S. Understanding of a convolutional neural network. In: *2017 International Conference on Engineering and Technology (ICET)*. 2017, s. 1–6. DOI: 10.1109/ICEngTechnol.2017.8308186.

[2] Boesch, G. *VGG very deep convolutional networks (vggnet) - what you need to know* [online]. Dec 2021. [Cit.: 2022-05-07]. Dostupné z: https://viso.ai/deep-learning/vgg-very-deep-convolutional-networks/.

[3] Chen, X., Bin, Y., Sang, N. a Gao, C. Scale Pyramid Network for Crowd Counting. In: *2019 IEEE Winter Conference on Applications of Computer Vision (WACV)*. 2019, s. 1941–1950. DOI: 10.1109/WACV.2019.00211.

[4] Choi, R. Y., Coyner, A. S., Kalpathy Cramer, J., Chiang, M. F. a Campbell, J. P. Introduction to Machine Learning, Neural Networks, and Deep Learning. *Translational Vision Science & Technology*. Únor 2020, sv. 9, č. 2, s. 14–14. DOI: 10.1167/tvst.9.2.14. ISSN 2164-2591. Dostupné z: https://doi.org/10.1167/tvst.9.2.14.

[5] Dike, H. U., Zhou, Y., Deveerasetty, K. K. a Wu, Q. Unsupervised Learning Based On Artificial Neural Network: A Review. In: *2018 IEEE International Conference on Cyborg and Bionic Systems (CBS)*. 2018, s. 322–327. DOI: 10.1109/CBS.2018.8612259.

[6] Dobeš, P., Špaňhel, J., Bartl, V., Juránek, R. a Herout, A. Density-Based Vehicle Counting with Unsupervised Scale Selection. In: *2020 Digital Image Computing: Techniques and Applications (DICTA)*. 2020, s. 1–8. DOI: 10.1109/DICTA51227.2020.9363401.

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

[8] Guerrero Gómez Olmedo, R., Torre Jiménez, B., López Sastre, R., Maldonado Bascón, S. a Oñoro Rubio, D. Extremely Overlapping Vehicle Counting. In: Paredes, R., Cardoso, J. S. a Pardo, X. M., ed. *Pattern Recognition and Image Analysis*. Cham: Springer International Publishing, 2015, s. 423–431. ISBN 978-3-319-19390-8.

[9] Hao, W., Yizhou, W., Yaqin, L. a Zhili, S. The Role of Activation Function in CNN. In: *2020 2nd International Conference on Information Technology and Computer Application (ITCA)*. 2020, s. 429–432. DOI: 10.1109/ITCA52113.2020.00096.

[10] 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.

[11] Hossain, M., Hosseinzadeh, M., Chanda, O. a Wang, Y. Crowd Counting Using Scale-Aware Attention Networks. In: *2019 IEEE Winter Conference on Applications of Computer Vision (WACV)*. 2019, s. 1280–1288. DOI: 10.1109/WACV.2019.00141.

[12] Hsieh, M.-R., Lin, Y.-L. a Hsu, W. H. *Drone-based Object Counting by Spatially Regularized Regional Proposal Network*. arXiv, 2017. DOI: 10.48550/ARXIV.1707.05972. Dostupné z: https://arxiv.org/abs/1707.05972.

[13] Ilyas, N., Ahmad, A. a Kim, K. CASA-Crowd: A Context-Aware Scale Aggregation CNN-Based Crowd Counting Technique. *IEEE Access*. 2019, sv. 7, s. 182050–182059. DOI: 10.1109/ACCESS.2019.2960292.

[14] Ioffe, S. a Szegedy, C. Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift. *CoRR*. 2015, abs/1502.03167. Dostupné z: http://arxiv.org/abs/1502.03167.

[15] Krizhevsky, A., Sutskever, I. a Hinton, G. E. ImageNet Classification with Deep Convolutional Neural Networks. In: Pereira, F., Burges, C., Bottou, L. a Weinberger, K., ed. *Advances in Neural Information Processing Systems*. Curran Associates, Inc., 2012, sv. 25. Dostupné z: https://proceedings.neurips.cc/paper/2012/file/c399862d3b9d6b76c8436e924a68c45b-Paper.pdf.

[16] Kurenkov, A. *A brief history of neural nets and deep learning* [online]. Sep 2020. [Cit.: 2022-05-07]. Dostupné z: https://www.skynettoday.com/overviews/neural-net-history.

[17] Lecun, Y., Bottou, L., Bengio, Y. a Haffner, P. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*. 1998, sv. 86, č. 11, s. 2278–2324. DOI: 10.1109/5.726791.

[18] Oñoro Rubio, D. a López Sastre, R. J. Towards Perspective-Free Object Counting with Deep Learning. In: Leibe, B., Matas, J., Sebe, N. a Welling, M., ed. *Computer Vision – ECCV 2016*. Cham: Springer International Publishing, 2016, s. 615–629. ISBN 978-3-319-46478-7.

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

[20] Saha, S. *A comprehensive guide to Convolutional Neural Networks - the ELI5 way* [online]. Towards Data Science, Dec 2018. [Cit. 2022-05-08]. Dostupné z: https://towardsdatascience.com/a-comprehensive-guide-to-convolutional-neural-networks-the-eli5-way-3bd2b1164a53.

[21] Sharma, S., Sharma, S. a Athaiya, A. Activation functions in neural networks. *Towards data science*. 2017, sv. 6, č. 12, s. 310–316.

[22] Simonyan, K. a Zisserman, A. *Very Deep Convolutional Networks for Large-Scale Image Recognition*. arXiv, 2014. DOI: 10.48550/ARXIV.1409.1556. Dostupné z: https://arxiv.org/abs/1409.1556.

[23] Sindagi, V. A. a Patel, V. M. A survey of recent advances in CNN-based single image crowd counting and density estimation. *Pattern Recognition Letters*. 2018, sv. 107, s. 3–16. DOI: https://doi.org/10.1016/j.patrec.2017.07.007. ISSN 0167-8655. Video Surveillance-oriented Biometrics. Dostupné z: https://www.sciencedirect.com/science/article/pii/S0167865517302398.

[24] Szegedy, C., Liu, W., Jia, Y., Sermanet, P., Reed, S. E. et al. Going Deeper with Convolutions. *CoRR*. 2014, abs/1409.4842. Dostupné z: http://arxiv.org/abs/1409.4842.

[25] Turkson, R., Yan, F., Ahmed Ali, M. a Hu, J. Artificial neural network applications in the calibration of spark-ignition engines: An overview. *Engineering Science and Technology, an International Journal*. Duben 2016, sv. 19. DOI: 10.1016/j.jestch.2016.03.003.

[26] Wang, J., Xiao, R., Guo, Y. a Zhang, L. Learning to Count Objects with Few Exemplar Annotations. *CoRR*. 2019, abs/1905.07898. Dostupné z: http://arxiv.org/abs/1905.07898.

[27] Zhang, L., Shi, M. a Chen, Q. Crowd Counting via Scale-Adaptive Convolutional Neural Network. In: *2018 IEEE Winter Conference on Applications of Computer Vision (WACV)*. 2018, s. 1113–1121. DOI: 10.1109/WACV.2018.00127.

[28] Zhou, X. Understanding the Convolutional Neural Networks with Gradient Descent and Backpropagation. *Journal of Physics: Conference Series*. IOP Publishing. apr 2018, sv. 1004, s. 012028. DOI: 10.1088/1742-6596/1004/1/012028. Dostupné z: https://doi.org/10.1088/1742-6596/1004/1/012028.

[29] Zhu, L., Zhao, Z., Lu, C., Lin, Y., Peng, Y. et al. Dual Path Multi-Scale Fusion Networks with Attention for Crowd Counting. *CoRR*. 2019, abs/1902.01115. Dostupné z: http://arxiv.org/abs/1902.01115.

[30] Šíma, J. a Neruda, R. *Teoretické otázky neuronových sítí*. Matfyzpress, 1996. 390 s. ISBN 80-85863-18-9.

Příloha A

Obsah přiloženého paměťového média

- `xvagne08.pdf` – text práce
- `src/` – složka obsahující zdrojové kódy
- `models/` – složka obsahující natrénované modely
- `readme.md` – návod k použití
- `plakát.pdf` – plakát shrnující tuto práci
- `video.mp4` – video shrnující tuto práci
- `latex.zip` – zdrojové soubory textu