Bakalářská práce

České
vysoké
učení technické
v Praze

F3

Fakulta elektrotechnická
Katedra kybernetiky

# Odhad rychlosti na segmentech silniční sítě

Martin Koryťák

Vedoucí: Ing. David Fiedler
Studijní program: Otevřená informatika
Studijní obor: Informatika a počítačové vědy
Květen 2018

# ZADÁNÍ BAKALÁŘSKÉ PRÁCE

## I. OSOBNÍ A STUDIJNÍ ÚDAJE

Příjmení: **Koryťák** Jméno: **Martin** Osobní číslo: **457010**

Fakulta/ústav: **Fakulta elektrotechnická**

Zadávající katedra/ústav: **Katedra kybernetiky**

Studijní program: **Otevřená informatika**

Studijní obor: **Informatika a počítačové vědy**

## II. ÚDAJE K BAKALÁŘSKÉ PRÁCI

Název bakalářské práce:

**Odhad rychlosti na segmentech silniční sítě**

Název bakalářské práce anglicky:

**Operating Speed Estimation on Road Segments**

Pokyny pro vypracování:

1. Prozkoumejte existující modely pro odvozování rychlosti při plynulé jízdě na různých segmentech silniční sítě.
2. Porovnejte vlastnosti modelů pro odvozování rychlosti a možnosti jejich aplikací.
3. Navrhněte řešení, které dokáže využít dopravní data, jako například záznamy jízdy autem, k přesnějšímu odhadu rychlosti.
4. Navržené řešení implementujte a experimentálně ověřte.

Seznam doporučené literatury:

[1] Transportation Research Board, "Modeling Operating Speed - Synthesis Report," Transportation Research Circulars, 2011.
[2] C. Heinze, M. Leodolter, H. Koller, D. Bauer. "Transferring urban traveling speed model fits across cities". European Transport Research Review, vol. 8 no. 3 pp. 19 2016

Jméno a pracoviště vedoucí(ho) bakalářské práce:

**Ing. David Fiedler, centrum umělé inteligence FEL**

Jméno a pracoviště druhé(ho) vedoucí(ho) nebo konzultanta(ky) bakalářské práce:

Datum zadání bakalářské práce: **08.01.2018** Termín odevzdání bakalářské práce: **25.05.2018**

Platnost zadání bakalářské práce: **30.09.2019**

| Ing. David Fiedler | doc. Ing. Tomáš Svoboda, Ph.D. | prof. Ing. Pavel Ripka, CSc. |
|---|---|---|
| podpis vedoucí(ho) práce | podpis vedoucí(ho) ústavu/katedry | podpis děkana(ky) |

## III. PŘEVZETÍ ZADÁNÍ

Student bere na vědomí, že je povinen vypracovat bakalářskou práci samostatně, bez cizí pomoci, s výjimkou poskytnutých konzultací. Seznam použité literatury, jiných pramenů a jmen konzultantů je třeba uvést v bakalářské práci.

. Datum převzetí zadání

Podpis studenta

# Poděkování

Rád bych poděkoval Ing. Davidu Fiedlerovi za cenné rady, věcné připomínky a vstřícnost při konzultacích bakalářské práce. Dále mé rodině, která mě po celou dobu studia podporovala a v neposlední řadě FEL ČVUT za poskytnuté vzdělání.

Tato práce vznikla za podpory projektů CERIT Scientific Cloud (LM2015085) a CESNET (LM2015042) financovaných z programu MŠMT Projekty velkých infrastruktur pro VaVaI.

# Prohlášení

Prohlašuji, že jsem předloženou práci vypracoval samostatně a že jsem uvedl veškeré použité informační zdroje v souladu s Metodickým pokynem o dodržování etických principů při přípravě vysokoškolských závěrečných prací.

. . . . . .

Martin Koryťák

V Praze dne 25. května 2018

# Abstrakt

Tato bakalářská práce se zabývá problémem predikce rychlosti vozidla na silničních segmentech. Rychlostní modely jsou v dnešní době důležité při plánování optimální trasy a v logistice, využití lze také nalézt v aplikacích jako je simulace dopravního provozu nebo při navrhování silničních úseků. Současné rychlostní modely využívají záznamy jízd, což tyto modely omezuje jen na silniční úseky, které jsou těmito daty pokryty. Existují také parametrické metody, které využívají většinou pouze jeden parametr silničního segmentu pro určení rychlosti. Díky dostupnosti satelitních a leteckých snímků byla navržena metoda, která využívá letecké snímky pro predikci rychlosti vozidel. V dnešní době se ke klasifikačním a regresním úlohám, které mají za datovou sadu obrázky, využívají konvoluční neuronové sítě. Pro řešení predikce rychlosti z leteckých snímků byly zvoleny čtyři state–of–the–art konvoluční neuronové sítě, kde byly použity váhy natrénované na datové sadě ImageNet. Kromě těchto sítí bylo experimentováno s hybridním modelem, který propojuje konvoluční neuronovou síť a parametrický model poskytnutý skupinou Smart Urban Mobility z Centra umělé inteligence FEL ČVUT. Nejlepší výsledek u konvoluční neuronové sítě dosáhl přesnosti průměrné absolutní chyby $9,11 \pm 0,025$ $km/h$. Hybridní model poté překonal model obsahující pouze konvoluční neuronovou síť, jeho přesnost byla $8,61 \pm 0,008$ $km/h$.



**Klíčová slova:** konvoluční neuronová síť, modelování rychlosti, hluboké učení, strojové učení, ImageNet, OpenStreetMap, Google Maps

**Vedoucí:** Ing. David Fiedler



# Abstract

This bachelor thesis focuses on operating speed estimation on road segments. Modeling operating speed is important in many applications such as planning optimal route, in logistics or designing road segment. Nowadays researchers use driving records to estimate operating speed on road segment. This method works only on road segments where driving records are available. There are also methods using mostly only one parameter of road segment to estimate operating speed. Due to great availability of satellite and aerial imagery, we suggest a novel method which uses aerial imagery for estimating operating speed. For machine learning problems whose dataset is composited of images, it is recommended to use convolutional neural network. We use four state–of–the–art convolutional neural network. We also take advantage of transfer learning to use features from a task with large dataset of ImageNet. Apart from state–of–the–art architectures, we experiment with hybrid model which concatenates convolutional neural network and parametric model developed by research group Smart Urban Mobility from Artificial Intelligence Center, CTU in Prague. The best mean absolute error among convolutional neural networks was $9,11 \pm 0,025$ $kph$. The best of all was hybrid model which achieved lower mean absolute error ($8,61 \pm 0,008$ $kph$) than any of convolutional neural networks.



**Keywords:** convolutional neural network, modeling operating speed, deep learning, machine learning, ImageNet, OpenStreetMap, Google Maps

**Title translation:** Operating Speed Estimation on Road Segments

# Obsah

# Obrázky



# Tabulky

# Kapitola 1
# Úvod

Tato práce se zabývá odhadováním rychlosti na segmentech silniční sítě. Predikce rychlosti nachází uplatnění v oblasti plánování optimální trasy a logistiky. Využití lze také nalézt v aplikacích jako je simulace dopravního provozu. Současné rychlostní modely většinou využívají záznamy jízd pro predikci rychlosti. Takto lze odhadovat rychlost jen pro konkrétní silniční segment, pro který jsou záznamy jízd k dispozici. Existují také parametrické modely, které odhadují rychlost vozidel většinou pomocí jednoho parametru silničního úseku. Protože letecké snímky jsou v dnešní době snadno dostupné, využili jsme je pro odhad rychlosti na silničním segmentu. Doporučenou metodou strojového učení pro úlohy, jejichž datovou sadu tvoří obrázky, je konvoluční neuronová síť. Tato metoda byla vybrána i pro náš problém odhadu rychlosti.

## 1.1 Motivace

Rychlost může být uvažována jako jeden z nejdůležitějších faktorů, které ovlivňují výběr trasy jízdy účastníků silničního provozu. Dále je považována za jedno z opatření, které konstruktéři silnic mohou využít při zkoumání očekávání a chování řidiče na silnicích [1]. Využití odhadování rychlosti vozidel je možné v oblastech jako je plánování optimální trasy nebo logistiky. Díky predikci rychlosti na silničních segmentech lze odhadnout dopravní situaci, která je potřebná při budování nové silniční infrastruktury nebo opravování stávající.

V současnosti existují dva hlavní typy rychlostních modelů. První typ využívá k určení rychlosti vozidla jednotlivé záznamy jízd pro daný silniční segment. Druhý typ odhaduje rychlost z geometrických, geografických i jiných vlastností tohoto segmentu. Výhodou prvního typu je snadné získání velkého počtu záznamů jízd ve městech. Naopak v méně osídlených oblastech, kde záznamy jízd zcela chybí nebo existují v malém počtu, je výhodné využít druhý typ odhadu rychlosti vozidla. Nevýhodou druhého typu predikce rychlosti bývá soustředění se na konkrétní typ silničního segmentu (například pouze na dvouproudé dálnice) a často malá datová množina.

## 1.2 Cíle a náplň práce

Prvním cílem této práce bylo prozkoumat existující modely pro odvozování rychlosti při plynulé jízdě na různých segmentech silniční sítě a vlastnosti modelů pro odvozování rychlosti a jejich možné aplikace. Prozkoumání modelů včetně jejich aplikace je popsáno v kapitole 2. Navržené řešení, které dokáže využít dopravní data k přesnějšímu odhadu rychlosti, je uvedeno v kapitole 3 spolu s popisem implementace. Provedené experimenty a jejich výsledky jsou v kapitole 4.

Celková práce je členěna do jednotlivých kapitol. Kapitola 2 se týká teoretického základu odhadovaní rychlostí vozidel současnými modely, který je důležitý k porozumění problematiky. Dále popisuje vývoj konvolučních neuronových sítí, běžně používané architektury a vrstvy. Kapitola 3 se zabývá výběrem metody a popisuje její implementaci. Kapitola 4 prezentuje a diskutuje dosažené výsledky. Kapitola 5 uzavírá tuto práci s krátkou úvahou dalšího rozšíření.

## 1.3 Definice použitých pojmů

Při popisu a analýze rychlostních modelů je třeba definovat několik pojmů. *Design Speed* je rychlost, která je určena k návrhu dané silnice. Je ovlivněna legislativou a může být nižší i vyšší než maximální povolená rychlost [2]. *Operating Speed* je nejvyšší rychlost, kterou se vozidlo může pohybovat za příznivého počasí a dopravních podmínek, aniž by překročilo bezpečnou rychlost určenou design speed [2]. *Posted Speed* neboli rychlostní limit, je dán legislativou v dané zemi nebo dopravním značením [2]. *85th Percentile Speed* je taková rychlost, kterou nepřekročí 85 procent řidičů na daném silničním úseku [2]. *Free–Flow Speed (FFS)* je rychlost vozidla na daném úseku, kterou se vozidlo pohybuje, pokud řidič není omezován dalšími vozidly nebo vnějšími vlivy jako je počasí [3].

Kromě definic typů rychlostí jsou důležité i další pojmy, které jsou spojeny se získáním dat pro odhadování rychlosti vozidla. *Floating Car Data (FCD)* označují použití dat generovaných jedním vozidlem jako vzorku pro posouzení celkového dopravního stavu. Tato data obvykle zahrnují základní telemetrii vozidla, jako je rychlost, směr a pozice vozidla [4]. *Trace* jsou jednotlivé FCD z jednoho vozidla, zachycují jízdu vozidla z bodu $A$ do bodu $B$. Z vizualizace je možné odhadnout, po kterých dopravních komunikacích se vozidlo přibližně pohybovalo. *Loop Detector* je zařízení, které je schopno detekovat přítomnost vozidla na silnici. Může být použito pro určení typu vozidla, jeho rychlosti nebo určování vytíženosti silnice. *Radar (měřící rychlost vozidla)* je zařízení, které je schopno určit rychlost vozidla.

Následující pojmy jsou běžně používané ve strojovém učení a jsou zde zmíněny pro snazší porozumění textu. *Outlier*, hodnota, která je velmi vzdálená svou polohou ostatním datům a tedy nekopíruje trend. *Ground Truth* je pojem, který znamená skutečný stav pozorovaného objektu.

# Kapitola 2
# Analýza problému

Tato kapitola se věnuje analýze problému odhadování rychlosti vozidel. Predikce rychlosti nachází uplatnění v oblasti plánování optimální trasy a logistiky, další využití lze nalézt v aplikacích jako je simulace dopravního provozu.

Modely, které využívají záznamy jízd, jsou uvedeny v podkapitole 2.1. Existují modely, které odhadují rychlost vozidel na základě vlastností silničního segmentu. Jejich aplikace a popis je v podkapitole 2.2. V poslední části této kapitoly 2.3 je popsáno využití počítačového vidění pro určování rychlosti.

## 2.1 Odhad rychlosti na základě naměřených cest na daném silničním segmentu

Odhadování rychlosti na základě naměřených cest na silnicích se v dnešní době stalo populární, a to díky poměrně levnému získávání dat (FCD) i faktu, že v současnosti je velké množství silnic pokryto záznamy o jízdě vozidel.

Data lze získat pomocí Global Positioning System (GPS), nebo je lze obdržet pomocí statických měřících zařízení jako jsou loop detectory. Z FCD se následně určuje směr jízdy a rychlost vozidla. Hlavní nevýhodou této metody je nevyvážené pokrytí v jednotlivých oblastech světa, zemích nebo městech. Ve větších městech nebo na dálnicích je těchto dat daleko více než v méně osídlených obcích či na okresních silnicích, proto v méně osídlených oblastech bude těžší korektně predikovat rychlost. Dále je možné, že FCD existují pro dané město nebo silniční úsek, ale není k nim přístup.

Pokud je na některém místě málo dat nebo neexistují žádná, může být tento problém řešen pomocí prostorové příbuznosti. Rychlost je tedy odvozena z rychlostí silnic, které vedou v protisměru nebo se na daný silniční segment napojují [5].

Někdy je tato oblast dále dělena na *path–based method* a *segment-based method*. *Segment–based method* nejprve odhaduje čas jízdy na každém úseku cesty zvlášť a nakonec všechny úseky sečte. Naproti tomu *path–based method* neuvažuje nutné časy pro zatočení nebo čekání vozidel na křižovatkách [6].

### 2.1.1 Používané metody

Základní metodou pro odhadování rychlosti vozidel na silnicích je *průměrování*. Výsledná hodnota je získaná jako průměr naměřených dob jízd ve stejný čas $t$ a určitý den $d$, jako je požadovaný čas.

Jednou z nejrozšířenějších metod strojového učení v oblasti predikce rychlosti je *lineární regrese*. V článku [7] byla data nasbírána z loop detectorů. Předpoklad autorů článku byl, že existuje lineární vztah mezi aktuálním časem jízdy a budoucím časem jízdy. Zde se jedná o online přístup odhadování rychlosti. Online přístup znamená, že odhad rychlosti vozidla je odvozen pouze z dat, která jsou aktuálně k dispozici. Autoři pracují s faktem, že data mají normální distribuci. Tento model ale vyžaduje, aby během jízdy mezi dvěma loop detectory nenastala žádná zásadní změna v dopravní situaci. Další nevýhodou této metody může být přílišný vliv outlierů. Autoři dosáhly při predikci vývoje rychlosti během jedné hodiny Root Mean Square Error (RMSE) v průměru 8 $km/h$. Tento výsledek se různil během denní doby, kdy během odpoledních hodin byla chyba vyšší než během nočních hodin.

Do kategorie regrese patří také *nelineární regrese*. Obvykle se jedná o polynomiální regresi nebo jako v článku [8], kde autoři použili loglineární regresi. Data byla rozdělena podle podobnosti jejich denních vzorců. Vyhodnocování následně probíhá na základě předešlých dní dané skupiny.

Další používanou metodou je *support vector regression*. Byla odvozena z metody support vector machines, která je velmi odolná proti outlierům. To je veliká výhoda například oproti lineární regresi. Míra ovlivňování outliery se dá regulovat. V článku [9] je prezentovaná tato metoda jako velmi úspěšná v porovnání s lineární regresí nebo průměrem z nasbíraných dat. Také je použita i na predikci doby jízdy několika silničních segmentů za sebou, kde předchozí uvedené modely většinou selhávaly. Nutno dodat, že tento model nevyužívá FCD, ale data z loop detectorů.

V posledním desetiletí se dostávají do popředí *neuronové sítě*. V článku [10] autoři používají údaje z nejvytíženějších denních hodin, které nasbírali v dopravních špičkách v pracovních dnech mezi druhou a osmou hodinou odpoledne. Jako výchozí metodu použili průměrování. Vstupem do neuronové sítě byla aktuální dopravní situace a průměrná rychlost vozidel. Jejich naučená neuronová síť daleko lépe predikovala dobu jízdy než výchozí metoda průměrování.

Všechny tyto články a metody popisují především krátkodobou predikci doby jízdy, tedy jejich modely zahrnují aktuální dopravní situaci na silnicích. Velký problém se objevuje v nemožnosti objektivního srovnaní metod, protože každý článek využívá jiná data a i přesnost modelu je dána za pomoci různých metrik.

## 2.2 Odhad rychlosti podle vlastností silničních segmentů

Odhad rychlosti podle vlastností silničních segmentů se snaží odhadnout rychlost vozidla na konkrétním typu silnice podle jejích parametrů. To je rozdíl oproti minulé kapitole, kde modely většinou odhadovaly rychlost vozidel pro všechny typy silnic v oblasti pokryté FCD.

Ve většině dostupných článků je k měření rychlosti, která je poté použita jako ground truth, použit radar měřící rychlost vozidla [11], [12]. Limitace v odhadování rychlosti za pomoci geometrických vlastností silnic je hlavně to, že tyto modely byly vytvořeny lidmi, kteří se zabývají především různými vlivy na rychlost a používají pouze jednoduché statistické metody. Autoři článků se většinou zaměřují pouze na jeden parametr silničního segmentu, který nejvíce ovlivňuje rychlost. Například ve článku [13] je zdůrazněno, že někteří autoři uvažují změnu rychlosti pouze v zatáčkách nebo při klesání a stoupání. Také měření probíhá pouze na předem vytypovaných silničních úsecích, tedy takové modely nejsou generalizující.

Ve většině článků je rychlost odhadovaná za pomoci horizontálních zakřivení silnice a maximální povolené rychlosti. Není zde bráno v potaz například to, kolik pruhů má daný silniční segment nebo zda se na tento silniční segment napojují další silnice. Mezi další příznaky, které mohly být uvažovány patří připojovací pruh (nebo jejich počet), pruh pro cyklisty, nebo autobusy.

### 2.2.1 Používané metody

V článku [11] se ukazuje, jak omezené jsou modely, které uvažují k odhadu rychlosti vozidla jen jednu geometrickou vlastnost silnice. Data byla nasbírána na 24 zakřivených úsecích a 36 rovných úsecích, přičemž rychlosti na rovných úsecích byly v intervalu od 48 $km/h$ do 88 $km/h$. Na zakřivených úsecích byly délky jednotlivých silničních segmentů omezeny na 200 $m$ před začátkem zakřivení. K získání dat byl použit radar měřící rychlost vozidla. Největší vliv na predikci rychlosti má rychlostní limit na silničním segmentu, který vysvětloval 53% rozptylu rychlosti. Bez uvažování rychlostního limitu vznikl model určený pouze šířkou jízdního pruhu, jednalo se o lineární regresi. Zajímavé je zjištění, že pokud se šířka jízdního pruhu zvětší o metr, vzroste predikovaní rychlosti o 15 $km/h$.

V dalším článku [12] je vybírán nejlepší příznak pro odhad rychlosti. Šířka a přítomnost krajnice, parkovací pruh nebo přechod pro chodce byly také vyhodnoceny, avšak nejvíce ovlivňujícím příznakem rychlosti vozidla je rychlostní limit na daných úsecích. Nutno podotknout, že každým příznakem byla rychlost jinak ovlivněna na rovných úsecích a na úsecích se zatáčkami. Zde je také zdůrazněno, že šířka silničního pruhu je podstatná pro odhadování rychlosti jen pro rovné úseky.

V článku [14] byla použita FCD z Vídně. Tato metoda sběru dat vyžaduje namapovaní daných tras na silniční síť a případně vyfiltrování outlierů za pomoci heuristik. Kromě lineární regrese, kde jako parametry autoři použili

typ silnice, maximální dovolenou rychlost a denní čas, zkoušeli i model, který uvažuje pouze maximální dovolenou rychlost. To podle očekávání nedopadlo nejlépe. Výsledný model byl naučený na datech z Vídně a otestován ve městě Linec. RMSE ve Vídni byl $8,30\ km/h$ a v Linci dosáhl $8,12\ km/h$.

Predikcí rychlosti v kopcovité krajině se zabývají autoři článku [15]. V článku je představena metoda jak pro predikci trajektorie vozidla, tak i metoda pro predikci rychlosti vozidla. Je ukázáno, že rychlost je závislá na trajektorii vozidla. Například pokud vozidlo nemusí přesně kopírovat zatáčku, dosáhne následně vyšší rychlosti. Predikce rychlosti je ovlivněna ohraničením silničního segmentu, tedy zda se na ní nachází chodník nebo jaká je krajnice. Dále je to maximální nebo minimální rychlost a nutnost vozidla se vyhýbat překážkám na silnici. Tento článek přináší výhody při navrhování silnic a silničních rychlostních limitů, protože ukazuje, které geometrické vlastnosti silničního segmentu nejvíce ovlivňují rychlost, stejně tak poskytuje výzkumníkům typické vzory chování řidičů na silnicích.

V článku [16] autoři naměřili rychlost pomocí radaru, kde silniční segment měl délku 6 200 $m$. Celkem bylo naměřeno 37 zakřivených úseků a 37 rovných úseků. U zakřivených úseků byl poloměr zatáčky mezi 18 $m$ a 148 $m$. Na každém úseku bylo měřeno alespoň 125 vozidel, která se pohybovala v intervalu od 37 $km/h$ do 73 $km/h$. Podle autorů jsou faktory, které ovlivňují 85th percentile speed, poloměr zakřivení silničního segmentu, stoupání nebo klesání.

#### 2.2.1.1 Parametrický model výzkumné skupiny Smart Urban Mobility

Zatím nepublikovaný model výzkumné skupiny Smart Urban Mobility z Centra umělé inteligence FEL ČVUT využívá čtyři parametry silničního segmentu pro odhad rychlosti vozidel – délka silničního segmentu, maximální povolená rychlost pro daný silniční úsek, počet pruhů silničního segmentu a kategorie silničního segmentu. Datovou množinu tvoří silniční síť Prahy. U některých silničních úseků bylo nutno za použití heuristik doplnit chybějící parametry. Uvedený model dosáhl průměrné absolutní chyby na validační sadě $9,41\ km/h$ a dále je použitý v hybridním modelu v této práci.

## 2.3 Počítačové vidění pro určování rychlosti

Aplikace počítačového vidění dokáží detekovat vozidla a monitorovat jejich jízdu. Z detekovaného vozidla lze určit rychlost vozidla, která poté slouží k vytvoření rychlostního modelu.

Použití počítačového vidění v oblasti určování rychlosti se dá rozdělit na dvě skupiny. První skupina se zabývá predikcí rychlosti na základě palubní kamery, tedy rozdílem mezi několika následujícími snímky. Druhá skupina předpovídá rychlost vozidel podle detekce vozidel a následného pohybu daného vozidla. Rozdíl mezi oběma metodami je v tom, že první přístup se snaží naučit vztah mezi několika následujícími snímky, tedy jak moc se snímek změnil

oproti předchozímu. Druhá metoda naopak detekuje vozidlo na dvou po sobě jdoucích snímcích a poté počítá vzdálenost mezi pixely, které charakterizují vozidlo. Následně je tato vzdálenost přepočtena pomocí času mezi pořízením snímků na skutečnou rychlost, jakou se pohybuje vozidlo zachycené na snímku.

V článku [17] se autoři zabývali tím, jak určit rychlost vozidla na základě dat z palubní kamery. Jako množinu dat k učení použili videozáznam, který byl rozdělen na 20 400 obrázků. Ground truth rychlostí rozdělili do skupin po čtyřech kilometrech za hodinu. Pro učení byly obrázky promíchány. Byla experimentálně vyzkoušena předučená FlowNet síť [18]. Jako lepší řešení se nakonec ukázalo použít VGG19 [19] síť naučenou na datové množině ImageNet [20], které byla změněna architektura do podoby architektury sítě FlowNet. Naučený model dosahoval 91 procent úspěšně klasifikovaných rychlostí. Tento model se naučil nezohledňovat jak silnici, tak i nebe. Zajímavostí je, že bral v potaz předek vozidla, což autoři vysvětlovali tím, že je to pro model signifikantní, protože se vozidlo ve vyšších rychlostech třese.

Zajímavý přístup je popsán v článku [21], kde autoři nejprve vytvoří z trasy vozidla v závislosti na čase snímek, který je poté vstupem do konvoluční neuronové sítě. Jelikož se jedná o neuronovou síť, autoři nemusí vybírat důležité vlastnosti ze snímku oproti [11] nebo [12]. Také zde dochází k dočasnému sdílení vah mezi jednotlivými neurony, tím se mohou šířit podstatné informace hlouběji do sítě.

V článku [22] autoři ukazují, že je důležité zvolit správný úhel, pokud odhadujeme rychlost vozidla z detekce a následného trasování vozidla. V tomto článku je popsáno, že rychlost se určuje pomocí dvou následujících kamerových snímků. Vzdálenost je vypočtena pomocí eukleidovské normy. Protože je kamera statická, dá se tedy určit, o kolik se posunulo detekované vozidlo v rámci jednoho snímku. Přesnost metody se pohybovala mezi 87 a 99 procenty. Ukázalo se, že pro rychlosti v intervalu od 15 $km/h$ do 20 $km/h$ je nejvhodnější pozorovací úhel kamery 45 stupňů, pro rychlosti z intervalu od 30 $km/h$ do 40 $km/h$ je nejvhodnější úhel 50 stupňů a pro rychlosti z intervalu od 50 $km/h$ do 60 $km/h$ je nejvhodnější úhel 60 stupňů. Největší chyba byla detekována na třídě rychlostí z intervalu 50 $km/h$ do 60 $km/h$.

Článek [23] se zabývá určením rychlosti z rozmazanosti obrázku. Tato metoda ovšem nefunguje, pokud rychlost vozidla je tak nízká, že rozmazání vozidla na snímku není pozorovatelné.

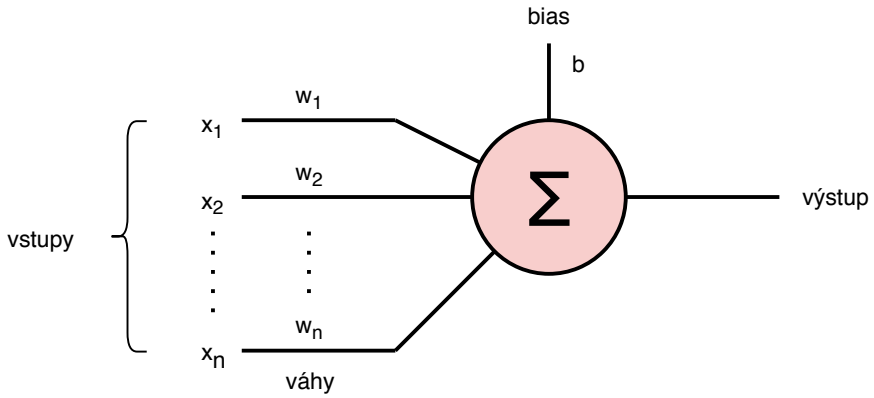
## 2.4 Konvoluční neuronové sítě a hluboké učení

Protože používáme pro odhad rychlosti letecké snímky, je důležité také popsat vybranou metodu. Doporučenou metodou strojového učení, pokud datovou sadu tvoří obrázky, je konvoluční neuronová síť.

Neuronové sítě jsou inspirovány biologickou analogií lidského mozku. Základní stavební jednotkou, stejně jako v mozku, je neuron viz Obrázek 2.1. Každý neuron dostane na vstup několik hodnot a vyprodukuje jeden výstup. Hlavní myšlenka je taková, že lze naučit sílu vlivu $w_i$ jednotlivých vstupů $x_i$. Tento základní model se nazývá perceptron.

Konvoluční neuronová síť se skládá z jedné nebo více konvolučních vrstev, po které následuje jedna nebo více plně propojených vrstev, jako ve standardní vícevrstvé neuronové síti. Architektura konvoluční neuronové sítě je navržena tak, aby využívala 2D strukturu vstupního snímku. To je dosažené pomocí lokálně propojených vah [24].

První známá konvoluční neuronová síť je LeNet [25]. Ta byla použita v úloze rozpoznávání ručně psaných číslic. V článku je dále ukázáno, že konvoluční neuronová síť v této úloze překonala svým výkonem klasickou neuronovou síť s plně propojenými vrstvami.




**Obrázek 2.1:** Výstup neuronu může být propojen a distribuován mezi další neurony, což může vytvořit komplexní neuronovou síť.

### 2.4.1 Datová množina ImageNet a soutěž ILSVRC

Neuronové sítě obecně potřebují velké množství vstupních dat pro jejich učení. Jednou z nejznámějších datových množin je ImageNet [20]. Ta obsahuje v současnosti okolo 14,2 miliónů anotovaných obrázků vysokého rozlišení různých kategorií (od druhů slonů po druhy dopravních prostředků).

Každoročně pořádaná soutěž ImageNet Large Scale Visual Recognition Competition (ILSVRC) slouží k porovnání nových výsledků na poli počítačového vidění. Tato soutěž je konaná na 1,2 miliónu obrázků z datové sady ImageNet, která pro klasifikaci obsahuje 1 000 skupin. Výzkumníci z celého světa soutěží v kategoriích segmentace a klasifikace objektů. Nejlepší sítě pak velmi často slouží jako state–of–the–art architektury v následujících letech.

### 2.4.2 Architektury sítí

V této části budou popsány důležité konvoluční neuronové sítě, které jsou využívané pro různé problémy počítačového vidění. Jednotlivé architektury jsou chronologicky řazeny tak, aby bylo patrné, jak se vyvíjí oblast konvolučních neuronových sítí.

Top–5 chyba je chyba, pokud ground truth není zahrnuta mezi pěti nejpravděpodobnějšími predikcemi sítě. Top–1 chyba je chyba, když predikce konvoluční neuronové sítě není shodná s ground truth.

#### 2.4.2.1 AlexNet

Síť [26] je tvořena pěti konvolučními vrstvami, max pooling vrstvami, dropout vrstvami a třemi plně propojenými vrstvami. Jako aktivační funkce byla použita ReLU 2.4.3.5, která snižuje čas potřebný k učení. Tato hluboká konvoluční neuronová síť vyhrála ILSVRC v roce 2012. Na testovací sadě dosáhla chyby top–5 15,3%. To byl znatelný úspěch oproti dosavadním state–of–the–art sítím. Při učení bylo potřeba 2 GPU s 3 GB paměti. Také bylo použito jen 1,2 miliónu obrázků z ImageNetu obsahující 1 000 různých kategorií.

#### 2.4.2.2 VGG16 a VGG19

Díky velmi malým konvolučním filtrům (3×3) mohli autoři [19] článku vytvořit 16 až 19 vrstev hlubokou neuronovou síť. Tento poznatek také velmi snížil počet parametrů generovaných každou vrstvou. V soutěži ILSVRC2014 tato síť vyhrála v úloze lokalizování a obsadila druhé místo v klasifikaci objektů. Proces trénování byl přejat od autorů AlexNetu. Síť VGG se stala velmi populární a i v dnešní době se velmi často využívá v úlohách počítačového vidění.

#### 2.4.2.3 GoogLeNet

Tato 22–vrstvá síť již neobsahuje pouze sekvenční strukturu a s výhodou využívá několik druhů vrstev paralelně. Autoři [27] zavedli tzv. „Inception module“, který používá konvoluční filtry o velikosti $1 \times 1$, $3 \times 3$ a $5 \times 5$. Ukázalo se, že je výhodnější nejprve začít s klasickou strukturou a tyto bloky použít až výše v síti. Důvodem bylo především velké zvyšování počtu výstupů mezi jednotlivými bloky a tím rostoucí paměťová náročnost. Výhodné bylo použití konvolučních filtrů o velikosti $1 \times 1$, které sníží dimenzionalitu v dimenzi filtrů. Také byla použita average pooling vrstva před klasifikací namísto plně propojené vrstvy, avšak dropout [28] zůstal ponechán. Díky tomu byla dosažena o 0,6% menší top–1 chyba. Autoři dále předpokládají, že k dosažení podobné kvality výsledků v úloze klasifikace a detekce by byla potřeba paměťově mnohem náročnější síť o podobné hloubce bez Inception bloků.

#### 2.4.2.4 ResNet

Autoři článku [29] představili reziduální bloky, které lze snadněji optimalizovat. Také si dokáží udržet nízkou složitost i při vysokém počtu vrstev a z ní získat lepší přesnost. Celý proces funguje tak, že k zobrazení $f(x)$ je ještě přičten

vstup $x$. Další výhodou je snadnější propagace gradientu při zpětné propagaci. S top–5 chybou $3,57\%$ vyhráli autoři soutěž ILSVRC2015.

#### 2.4.2.5 DenseNet

Hlavní myšlenka DenseNet sítě [30] je založena na propojení vrstev s každou další vrstvou v dopředném směru. Z této myšlenky plynou výhody jako je zmírnění problému mizícího gradientu nebo výrazné snížení počtu parametrů. DenseNet podává konzistentní zvyšování přesnosti při zvyšování počtu parametrů aniž by bylo znatelné přeučení. K dosažení obdobné přesnosti jako state–of–the–art sítím DenseNet síti stačí menší počet parametrů a méně výpočtů. Na datové množině ImageNet tato síť dosáhla top–5 chybu $6,12\%$.

### 2.4.3 Používané vrstvy

Neuronové sítě se skládají z jednotlivých neuronů a existuje velké množství jejich propojení a vrstvení. Konvoluční neuronové sítě využívají faktu, že na vstupu jsou většinou obrázky, proto jednotlivé vrstvy mají tři rozměry – výšku, šířku a hloubku. Obrázky se skládají ze tří barevných kanálů (červený, zelený a modrý). Každý pixel je reprezentován třemi čísly (pro každý kanál jedno číslo) z intervalu 0 až 255. Při skládání jednotlivých vrstev na sebe se používá nejvíce konvoluční, pooling a plně propojená vrstva.

#### 2.4.3.1 Konvoluční vrstva

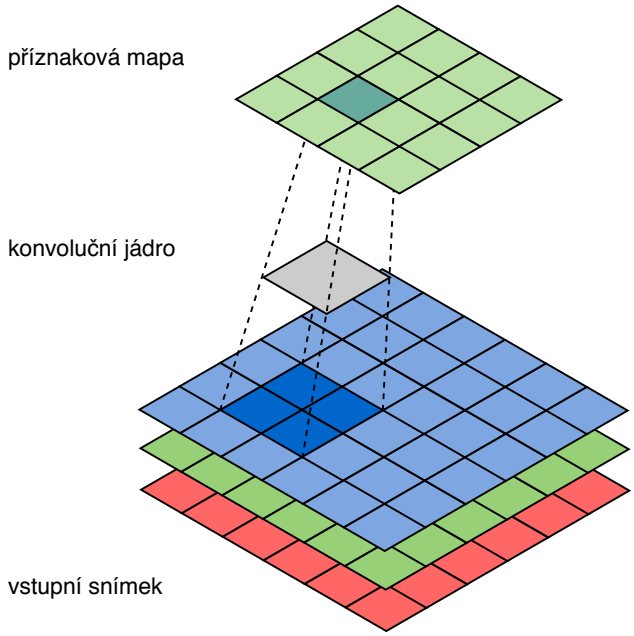
Jedná se o základní vrstvu, která tvoří konvoluční neuronovou síť. Každý neuron sleduje pouze určitou část vstupního obrázku, díky čemuž značně redukuje počet výstupů. Vzhledem k této vlastnosti je konvoluční vrstva použita v první části sítě, která slouží především k extrahování příznaků viz Obrázky 2.7 a 2.2. Nejdůležitější parametr konvoluční vrstvy je velikost konvolučního jádra a počet příznakových map.

#### 2.4.3.2 Pooling vrstva

Pooling vrstva se vkládá mezi jednotlivé konvoluční vrstvy kvůli zmenšení počtu parametrů. To má za následek lepší generalizaci modelu. Pooling vrstvy mohou být různého typu (max pooling, average pooling) jako na Obrázku 2.3. Může být ovšem vložena i místo plně propojené vrstvy na konec sítě, jak zmiňují články [27] a [31].

#### 2.4.3.3 Plně propojená vrstva

Tato vrstva se skládá z jednotlivých neuronů, které jsou propojeny s každým výstupem aktivační funkce předchozí vrstvy. Kvůli generování velkého množství parametrů se nachází až na úplném konci konvoluční neuronové sítě. Navíc učení sítí, které obsahují pouze plně propojené vrstvy (Obrázek 2.4), je obtížné a také náchylné k přeučení.




**Obrázek 2.2:** Filtr konvoluje přes vstupní snímek, který je reprezentován tříroz-měrným polem hodnot.

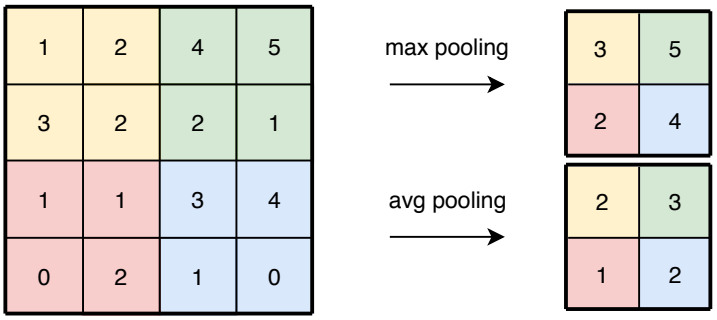
#### 2.4.3.4 Dropout

Dropout [28] je doporučován jako metoda, která efektivně zabraňuje přeučení. Hlavní myšlenka je v náhodné deaktivaci neuronů v neuronové síti během učení. To zabrání neuronům v přílišné adaptaci na vstupy. Ve fázi testování sítě jsou vždy všechny neurony aktivní. Jejich váhy jsou však pronásobeny pravděpodobností deaktivace neuronů $p$. Nejčastěji se dropout vkládá mezi dvě plně propojené vrstvy. To zapříčiní, že po aplikaci dropoutu se plně propojená vrstva ztenčí viz Obrázek 2.5.

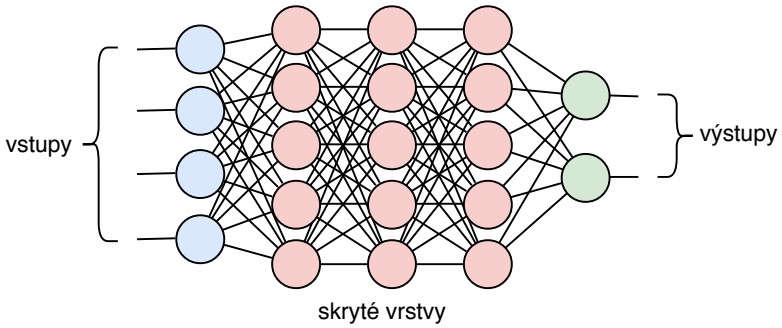
Kromě dropoutu se k zabránění přeučení používají ještě L1 norma, která ke ztrátové funkci přičte absolutní hodnotu každé váhy vynásobenou konstantou $\Lambda$. Konstanta $\Lambda$ vyjadřuje velikost normalizace. Naopak L2 norma přičte druhou mocninu vah vynásobenou konstantou $\frac{1}{2}\Lambda$ ke ztrátové funkci [32]. Někdy se obě normy kombinují viz [33].

#### 2.4.3.5 Aktivační funkce

Aktivační funkce provede určitou matematickou operaci s číslem, které obdrží na vstupu – například na rovnici 2.1 je zachycena ReLU aktivační funkce. Historicky volenou aktivační funkcí je sigmoid aktivační funkce, ta v dnešní době již není v oblasti neuronových sítí používaná kvůli saturaci. To znamená, že gradient je téměř nulový, což je pro zpětnou propagaci nežádoucí. Podobnou aktivační funkcí je hyperbolický tangent. Oproti sigmoid aktivační funkci má výhodu středu v počátku kartézského souřadnicovém systému. V poslední době se používá ReLU aktivační funkce nebo její modifikace [32]. Na Obrázku 2.6




**Obrázek 2.3:** Pooling vrstva snižuje velikost vstupu, častěji používaná je max pooling vrstva. Ta efektivně sníží počet parametrů v síti a tím přispěje ke snížení rizika přeučení.




**Obrázek 2.4:** Klasická neuronová síť se skládá pouze z plně propojených vrstev. Tyto vrstvy generují velké množství parametrů, které se síť dokáže naučit.

jsou vykresleny všechny aktivační funkce použité pro řešení problému odhadu rychlosti na silničních segmentech.

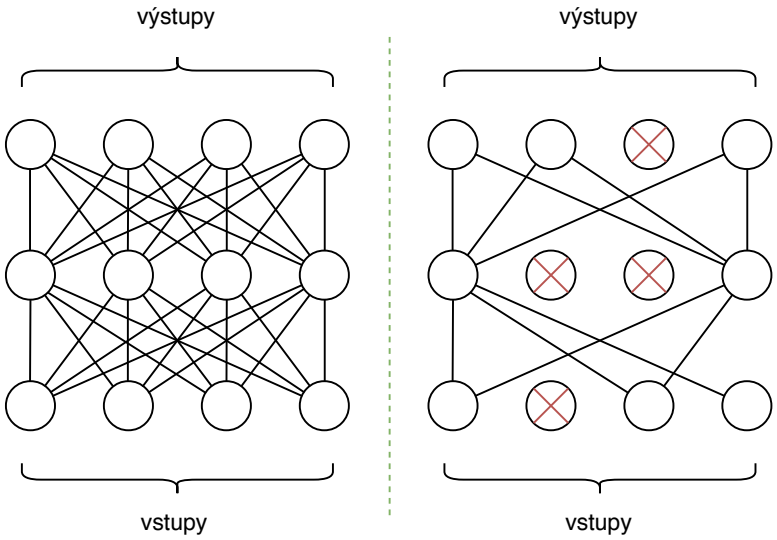
$$výstup = \max(0, bias + \sum_{j=1}^{n} váha_j \cdot vstup_j) \tag{2.1}$$

### 2.4.4 Obvyklá architektura konvoluční neuronové sítě

Při návrhu sítě se používají především konvoluční, pooling a plně propojené vrstvy. Následující schéma 2.7 ukazuje obvyklou praxi při vrstvení doporučovanou výzkumníky zabývajícími se konvolučními neuronovými sítěmi. Takové schéma lze využít pro úlohy, pro které zatím neexistuje obecné doporučení sítě.

## 2.5 Hluboké učení z leteckých a satelitních snímků

Dostupnost satelitních a leteckých snímků země v dnešní době umožňuje velké množství výzkumu v této oblasti. Dalším usnadněním pro výzkum je pokrytí satelitními snímky celého zemského povrchu. Data jsou získávána ze satelitů



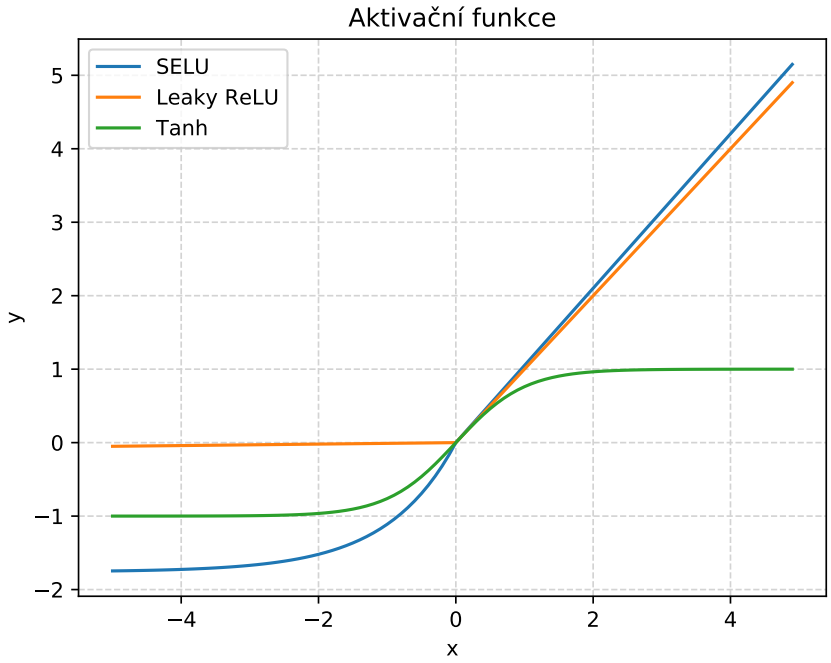

**Obrázek 2.5:** Vlevo síť s plně propojenými vrstvami, vpravo pak po aplikaci dropoutu s pravděpodobností $p$.

nebo letadel s různým prostorovým rozlišením a natočením [34]. Díky tomu lze poté modelovat 3D povrch země. Dříve se jednalo například o segmentaci objektů, nyní se jedná převážně o klasifikaci a detekci objektů v leteckých snímcích nebo monitorování vegetace a zemědělství [35]. Klasifikování snímků do sémanticky podobných kategorií je hlavním cílem v oblasti rozpoznávání objektů z leteckých snímků. Toto je pro vědce velkou motivací, protože na zemském povrchu se mohou objevit různé objekty v rozličných velikostech a natočení. Proto je správná detekce nebo klasifikace obtížná.

Jedním z hlavních problémů, kterým vědci v dnešní době čelí, je nedostatečně bohatá oštítkovaná datová množina. Získávání ground truth pro letecké nebo satelitní snímky je velmi obtížné viz [36].

Autoři článku [34] se snažili klasifikovat různé objekty na satelitních snímcích jako jsou letadla, lodě nebo stadióny. Ve svém výzkumu využili již předučené konvoluční sítě CaffeNet [37] a GoogleLeNet [27]. Nejlepší výsledek byl dosažen při použití GoogleLeNet, kdy přesnost klasifikace byla 97,1 procenta. Při bližším zkoumání naučené GoogLeNet sítě bylo zjištěno, že nejobtížnější bylo správně určit třídu „husté osídlení“, což může být zapříčiněno přítomností velmi podobných tříd jako jsou třídy „středně hustě osídleno“ nebo „parkoviště obytných vozů“.

V článku [35] je snaha dokázat, že lze naučit konvoluční neuronovou síť klasifikovat snímky podle jednotlivých pixelů. Autoři ručně přiřadili třídu každému pixelu na satelitním snímku. Každý pixel přitom odpovídá půl metru ve skutečnosti. Kvůli nedostatku dat byly třídy „železnice“ a „parkoviště“ sjednoceny s třídou „silnice“. Jednotlivé filtry v konvoluční neuronové síti jsou nejprve učeny pomocí algoritmu K–means. To umožňuje učení filtrů přímo z dat bez použití ground truth. Přesnost klasifikace při použití jedné



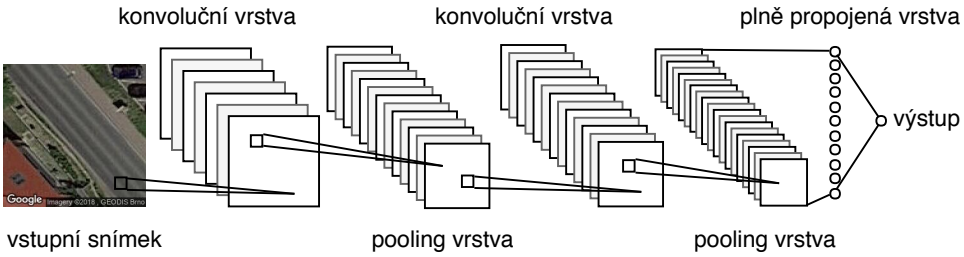

**Obrázek 2.6:** Aktivační funkce ReLU podle [26] 6–krát rychleji konverguje než hyperbolický tangent. Leaky ReLU i SELU jsou modifikované verze ReLU.

konvoluční neuronové sítě byla 90 procent, při kombinaci více konvolučních neuronových sítí byla dosažena přesnost 94,5 procent.

Odhadem chudoby afrických zemí ze satelitních snímků se zabýval článek [38]. Autoři použili satelitní snímky pořízené za denního světla, tedy jejich model neuvažuje ovlivnění chudoby na základě nočních osvětlení obcí nebo dat z mobilních telefonů. Model vychází z předučené konvoluční neuronové sítě na datové množině ImageNet. Model má vysokou přesnost, přestože data obsahují nepřesné údaje o čase pořízení snímku v trénovacích datech. Model vysvětluje až 55% rozptylu spotřeby peněz v průměrné domácnosti a až 75% rozptylu bohatství aktiv v průměrné domácnosti.

Autoři článku [39] se snaží o detekci vozidel na satelitních snímcích. Přičemž jejich metoda se snaží kromě detekce vozidel, detekovat i směr jízdy vozidla. Detekce vozidel v satelitních snímcích poskytuje informace o dlouhodobém stavu dopravy a umožňuje ji lépe plánovat. Detekci vozidel komplikuje často malé rozlišení snímků. Datová množina snímku byla pořízena z výšky jednoho kilometru s 21–Megapixelovým fotoaparátem. Snímky byly ručně oštítkovány a ukázalo se, že třída kamión obsahuje malé množství záznamů, proto bylo nutné tuto třídu uměle zvětšit. Protože metoda detekce vozidel je rychlá, je možné ji využít i v real–time aplikacích.

V článku [40] se autoři zabývají detekcí silnic ze satelitních snímků. Přitom klasický přístup [41] detekce silnic ze satelitních snímků je detekce hran silničního segmentu, který je v kontrastu s okolím. V článku se ukazuje, že




**Obrázek 2.7:** Obvyklé schéma při návrhu konvoluční neuronové sítě.

nestačí pouze část silnice, ale je zapotřebí i kontext, tedy je nutné na snímku zachytit i nejbližší okolí silničního segmentu, aby síť dosáhla požadované přesnosti. Síť byla inicializovaná za pomoci učení bez učitele, což se ukazuje jako velké zlepšení oproti náhodné inicializaci [42]. Výstup modelu je pravděpodobnost „silnice nebo ostatní“ pro každý pixel zvlášť. Výhoda oproti ostatním modelům je v učení sítě na GPU a také v podstatně větší datové množině. Model byl otestován na dvou datových množinách, kde úspěšná detekce silnice se lišila na každé sadě. To může být způsobeno například odlišným plánováním a stavbou silnic v obou datových množinách.

# Kapitola 3
# Metodologie

Díky snadné dostupnosti leteckých a satelitních snímků, lze tato data využít v aplikacích jako je určování chudoby afrických zemí [38] nebo detekce silnic [40]. Pro úlohy, jejichž datovou množinu tvoří letecké nebo satelitní snímky, je v dnešní době state–of–the–art metodou strojového učení konvoluční neuronová síť. Čtyři state–of–the–art konvoluční neuronové sítě byly použity při řešení úlohy odhadu rychlosti na silničních segmentech.

V této kapitole je nejprve popsán soubor obsahující rychlosti a silniční síť 3.1, poté je popsána tvorba datové množiny 3.2, která se skládá z leteckých snímků silničních segmentů. Dále je uveden postup umělého zvětšování datové množiny pomocí transformací snímků 3.3. Jako metoda k řešení postupu byla vybrána konvoluční síť. Popis hyperparametrů a postup učení sítě je popsán v sekcích 3.4 a 3.5. Následně jsou zmíněny zbylé detaily nutné k pochopení celkového postupu, jehož implementace je popsaná v poslední části kapitoly.

## 3.1 Data o rychlosti

Vstupní soubor obsahující silniční síť Prahy s rychlostmi vozidel na silničních segmentech byl poskytnut skupinou Smart Urban Mobility z Centra umělé inteligence FEL ČVUT. Ten obsahuje seznam hran představující silniční úseky, které jsou uloženy ve formátu GeoJSON [1]. Každá hrana reprezentuje skutečný silniční úsek na území Prahy. Každá hrana obsahuje dvě GPS souřadnice (začátek a konec), unikátní identifikační číslo a další hodnoty jako například naměřenou rychlost na daném silničním segmentu. Ukázku vstupního souboru zachycuje Obrázek 3.1.

Seznam hran představuje skutečnou silniční síť Prahy. To může být využito pro doplnění dalších hodnot jako je převýšení nebo délka silničního úseku, které mohou být využity při tvorbě hybridního modelu.

[1] http://geojson.org

```
{
   ''features'': [
      {
         ''geometry'': {
            ''coordinates'': [
               [
                  14.4080202,
                  50.0355033
               ],
               [
                  14.4080585,
                  50.035151
               ]
            ],
            ''type'': ''LineString''
         },
         ''properties'': {
            ''highway'': ''trunk'',
            ''id'': 0,
            ''lanes'': 3,
            ''maxspeed'': ''80'',
            ''measured_speed'': 77.82634766277053,
            ...
         },
         ''type'': ''Feature''
      },
      ...
   ],
   ''type'': ''FeatureCollection''
}
```


**Obrázek 3.1:** Ukázka formátu GeoJSON s hranami reprezentujícími silniční síť v Praze.

## 3.2 Získávání leteckých snímků

Celý vstupní GeoJSON soubor byl vygenerován z OpenStreetMap[2] (OSM). To znamená, že byla vybrána a stažena část Středočeského kraje zahrnující Prahu. Po stažení nejprve proběhne vyfiltrování pouze veřejně dostupných ulic a poté je vyfiltrovaný soubor převeden do formátu GeoJSON. Data z OSM s sebou přináší úskalí v tom, že GPS souřadnice neodpovídají přesně souřadnicím z Google Maps. Přestože jsou letecké snímky centrovány na silniční segment, nemusí silnice na získaném snímku procházet přesně středem. Mezi další nevýhody se řadí neúplné informace u jednotlivých silničních segmentů v datech z OSM. Velmi často chybí informace o maximální povolené rychlosti nebo počtu pruhů. Tento údaj bývá často chybně vyplněný, což znesnadňuje používání těchto dat. Pro parametrický model byla potřeba některé údaje vyplnit za použití několika heuristik. Často chybějící údaj o maximální povolené

[2] https://www.openstreetmap.org

rychlosti byl určen z typu silničního segmentu. Tato jednoduchá heuristika přiřadí dálnicím maximální povolenou rychlost $v = 130\ km/h$, obytným zónám $v = 20\ km/h$ a ostatním typům silničních segmentů $v = 50\ km/h$ (vzhledem k tomu, že jsou zohledňované silniční segmenty na území Prahy). Délka silničního úseku je exaktně určena pomocí GPS souřadnic a Haversinova vzorce.

Maps Static API[3] od společnosti Google nabízí přístup k leteckým snímkům země a dalším údajům. Ukázka volání viz Obrázek 3.2, vyžaduje několik parametrů. Ty, které byly skutečně použity jsou uvedeny níže:

- **center:** vyžaduje GPS souřadnice, které budou uprostřed vráceného snímku
- **size:** udává rozměry požadovaného snímku v pixelech
- **scale:** určuje počet pixelů vrácených voláním
- **zoom:** udává míru přiblížení vzhledem k zemskému povrchu
- **maptype:** umožňuje volbu typu požadované mapy
- **format:** určuje formát snímku, který má být dotazem vrácen
- **key:** prostor pro vložení unikátního API klíče

```
https://maps.googleapis.com/maps/api/staticmap?center=50.0584491%
2C14.51040565&maptype=satellite&key=API_key&scale=1&format=jpg&
size=224x224&zoom=20
```

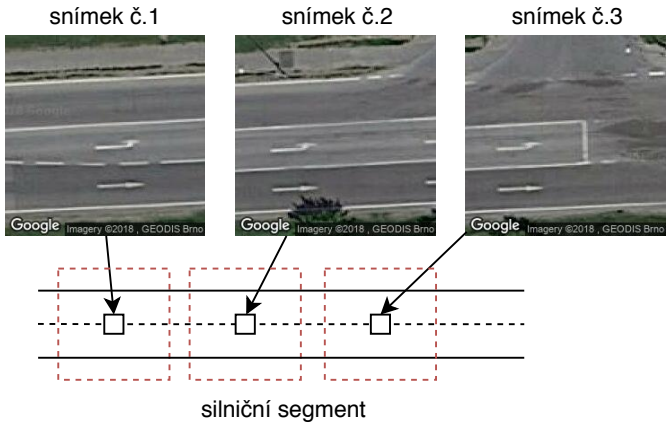
**Obrázek 3.2:** Ukázka volání Google Maps Static API.

S využitím Google Maps Static API byly vytvořeny dvě různé datové množiny, které obsahují letecké snímky silničních segmentů. Obě datové sady obsahují čtvercové snímky o rozměru 224 pixelů. Tato velikost byla inspirována většinou state–of–the–art modely, které byly učeny na obrázcích stejné velikosti. Další motivací poté byla velikost datové množiny v úložišti a rychlost učení sítí.

První datová množina (CNTR50) je tvořena snímky pořízenými vždy uprostřed silničního segmentu, tedy jeden obrázek charakterizuje jeden silniční úsek. Druhá datová množina (MAX100) je tvořena tak, že silniční segment je rozdělen na menší úseky tak, aby se získané snímky téměř nepřekrývaly a tedy obsahovaly co nejvíce informací 3.3. To ovšem není vždy splněno kvůli mírně rozdílným GPS souřadnicím OSM a Google Maps, jak bylo zmíněno výše.

Poslední datová množina (AUG200) byla vytvořena augmentací snímků z první datové množiny. Podrobněji popsaný postup viz sekce 3.3.

[3] https://developers.google.com/maps/documentation/maps-static/intro




**Obrázek 3.3:** Rozdělení silničního segmentu na části podle požadované velikosti snímků. Každý snímek je vycentrován na střed dané části silničního segmentu.

## 3.3 Augmentace obrázků

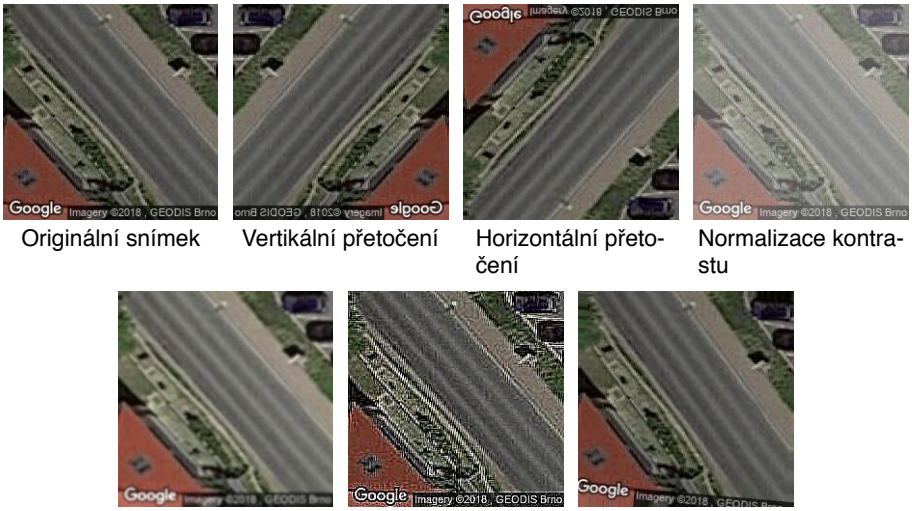
Pokud má konvoluční neuronová síť velké množství parametrů, je poté schopná si zapamatovat všechna trénovací data. Tento jev se nazývá přeučení sítě a projevuje se velmi malou trénovací chybou, ale poměrně vysokou chybou na testovacích datech.

Jednou z účinných metod proti přeučení složité konvoluční neuronové sítě a malé datové množiny je augmentace dat. Hlavní myšlenka spočívá v umělých transformacích obrázků tak, aby se zvýšil jejich počet a síť začala lépe generalizovat. Tyto transformace ovšem nemohou být libovolné, protože by mohly zmást konvoluční neuronovou síť. Například u klasifikace zvířat nelze použít horizontální přetočení. U snímků silnic lze tuto možnost použít vzhledem k tomu, že úsek silniční sítě je zachycen z ptačí perspektivy. Vzhledem ke konstantní výšce pořízení satelitních snímků nelze použít přiblížení ani zastřižení. Stejně tak i posunutí není možné použít, protože silniční segment prochází středem snímků. Použité transformace jsou níže uvedeny, popsány a zobrazeny na Obrázku 3.4.

Pro každý původní obrázek byly vytvořeny čtyři nové umělé obrázky, na které byly aplikovány transformace v náhodném pořadí. Počet transformací pro každý snímek byl také náhodně určen. Hodnoty pro jednotlivé parametry transformací jsou inspirovány [43] a [36]. U rotace bylo experimentálně zjištěno, že při větším natočení se výrazně změní informace na snímku. Změna informace je zapříčiněna umělým doplněním chybějících pixelů.

### 3.3.1 Přetočení

Použité přetočení bylo jak horizontální, tak i vertikální. Přetočení nezmění rozměry snímku ani hodnoty pixelů, pouze dojde k jejich přesunu.




**Obrázek 3.4:** Ukázka všech použitých transformací, které byly aplikovány na datovou množinu.

### 3.3.2 Rotace

Rotace snímku o náhodný počet stupňů mezi 5 a -5 stupni. Při rotaci zůstávají nevyplněné pixely, ty jsou uměle doplněny hodnotami krajních pixelů.

### 3.3.3 Normalizace kontrastu

Kontrast udává rozpětí mezi světlými a tmavými pixely obrázku. Změna kontrastu obrázku je určena pomocí parametru $\alpha$. Byly použity hodnoty v rozmezí 0,5 až 1,1.

### 3.3.4 Gaussovské rozostření

Gaussovo rozostření nastaví hodnotu každého pixelu v obrázku na průměrnou hodnotu pixelů v okolí $\sigma$. Vyšší poloměr znamená větší rozostření. Velikost okolí byla stanovena mezi 0,25 a 1,3.

### 3.3.5 Ostření

Při ostření obrázku byly nastavovány hodnoty alfa $\alpha$ a světlost $\psi$. Alfa určuje viditelnost zaostřeného a původního obrázku. Pokud je $\alpha = 0$, pak pouze původní obrázek je viditelný, naopak $\alpha = 1$ znamená viditelnost pouze zaostřeného obrázku. Světlost obrázku byla vybrána z rozmezí 0,75 až 1,5 a alfa byla stanovena z hodnot od 0,1 do 0,8.

## 3.4 Hyperparametry konvoluční neuronové sítě

Ve strojovém učení je hyperparametr takový parametr, jehož hodnota je nastavená ještě před fází učení. Hodnota hyperparametru se nemění během fáze učení. Tyto hyperparametry mohou ovlivnit jak dobu učení, tak i celkovou přesnost algoritmů strojového učení.

Neuronové sítě mají obecně velké množství hyperparametrů. Mezi nejdůležitější patří *learning rate*, počet a typ vrstev, *batch velikost*, míra regularizace a aktivační funkce. Learning rate určuje velikost změny vah v síti a batch velikost je počet dat, které jsou použity pro jednu aktualizaci vah sítě.

Optimalizace hyperparametrů vyžaduje kromě trénovací množiny také validační množinu. Validační množina obsahuje data, která nebyla použita pro učení a měří se na ní přesnost sítě (podrobněji popsáno v sekci 3.5.2). Celý proces optimalizace hyperparametrů zahrnuje trénování sítě na vybraných hyperparametrech a podle validační chyby se určí, zda hyperparametry byly vhodně zvoleny. Pro přesnější vyhodnocení se používá křížová validace, která proces učení výrazně zpomalí.

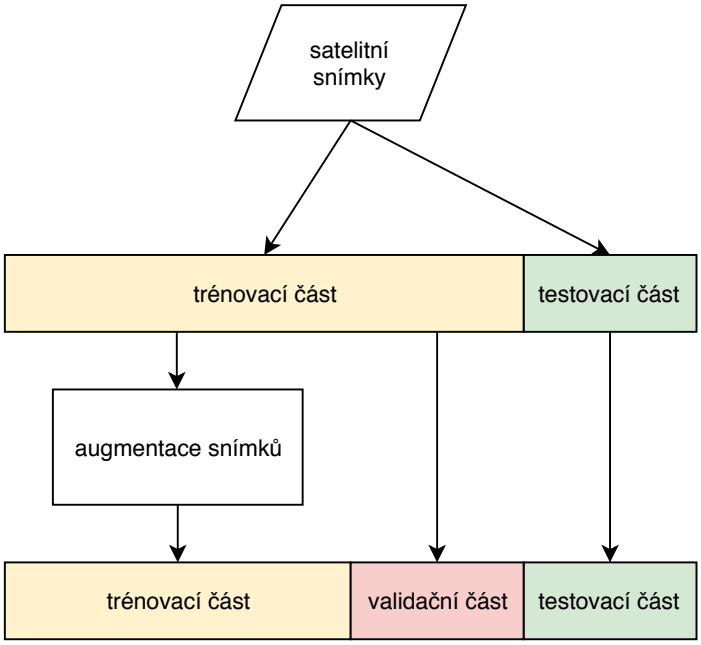
Na optimalizaci hyperparametrů existuje několik algoritmů [44]. Naivní přístup představuje grid search, který vyhodnocuje pouze předem definované kombinace hyperparametrů viz [45]. Random search [46] umožňuje vyšší výkon i větší prohledání prostoru hyperparametrů. Je upřednostňován i před Bayesovskou optimalizací například v kurzu [32].

## 3.5 Učení sítí

V této sekci jsou uvedeny všechny použité metody pro učení konvolučních sítí. Popsána je i metoda rozdělení dat a křížová validace.

### 3.5.1 Rozdělení dat

Tradičně je používáno rozdělení dat na tři části – trénovací, validační a testovací. Většinou stačí data náhodně rozdělit mezi tyto tři části. Každá z částí by měla mít podobnou distribuci jako celá datová množina. Rozdělení dat pro datové množiny MAX100 a AUG200 je díky charakteru stažených leteckých snímků komplikovanější. Datová množina MAX100 (viz 3.2) obsahuje pro jeden úsek více snímků, proto je nutné vložit snímky z jednoho silničního segmentu pouze do jedné z částí. Protože může být v trénovací sadě obsažen alespoň jeden snímek pro většinu silničních segmentů, čímž může dojít k přeučení sítě, které by navíc nebylo odhaleno ani při závěrečném testování. Pro augmentovaná data platí, že by měla být použita pouze pro účely trénovaní, a tedy tato data nesmí být obsažena ani ve validační ani testovací sadě viz Obrázek 3.5.

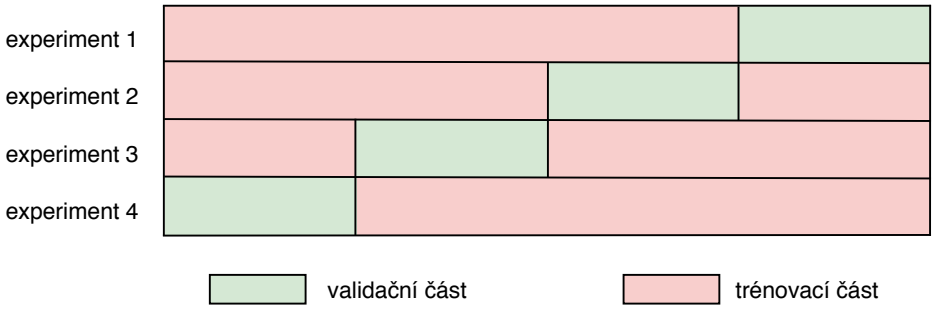



**Obrázek 3.5:** Popis rozdělení dat na trénovací, validační a testovací část. Při rozdělení dat na trénovací a validační část nesmí být augmentovaná data obsažena ve validační části.

### 3.5.2 Křížová validace

Pro relevantnější výsledky byla použita křížová validace viz Obrázek 3.6. Datová množina se nejprve rozdělí na podmnožiny. Jedna z nich je poté použita k vyhodnocení sítě, zatímco zbylé podmnožiny jsou použity pro učení. Tento proces se opakuje pro všechny kombinace rozdělení.

Protože optimalizace hyperparametrů je početně náročná, byla použita nejmenší datová množina CNTR50 obsahující 50 855 leteckých snímků. Kvůli tomu byla použita 3–násobná křížová validace, aby se zabránilo přílišnému ovlivňování nezávislými vzorky dat. Při učení sítě byla použita 5–násobná křížová validace.




**Obrázek 3.6:** Schéma 4-násobné křížové validace.

### 3.5.3 Ztrátová funkce a volba optimalizačního algoritmu

Pro řešení daného problému byl zvolen algoritmus zpětného šíření. Protože se jedná o regresní úlohu, kde síť se snaží minimalizovat rozdíl mezi skutečnou rychlostí na silničním segmentu a její predikcí, je ztrátová funkce nejčastěji jedna z rovnic 3.1 a 3.2. Druhá rovnice 3.2 podle [32] je preferovaná a snadněji optimalizovatelná. Také je pro daný problém důležitější průměrná absolutní chyba než průměrná druhá mocnina chyby, protože je odolnější vůči outlierům.

K optimalizování ztrátové funkce bylo experimentováno se třemi optimalizačními algoritmy – Nadam, RMSProp a SGD. Optimalizační algoritmy slouží k nalezení vhodného (optimálního) řešení problému zejména v případech, kdy není znám matematický popis řešení problému [47]. Nadam a RMSProp jsou adaptivní optimalizační algoritmy, zatímco SGD nabízí větší kontrolu nad zvolenými parametry jako je learning rate 3.4 nebo momentum, které snižuje riziko uváznutí v lokálním minimu a zvyšuje rychlost konvergence.

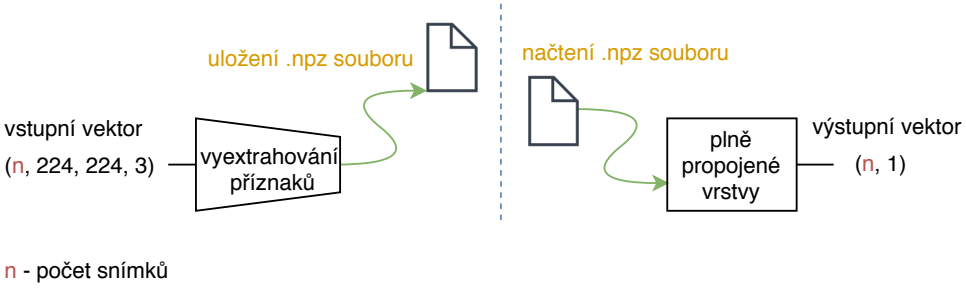
$$MSE = \frac{1}{n}\sum_{i=1}^{n}(rychlost_i - predikovaná\_rychlost_i)^2 \tag{3.1}$$

$$MAE = \frac{1}{n}\sum_{i=1}^{n}|rychlost_i - predikovaná\_rychlost_i| \tag{3.2}$$

### 3.5.4 Extrahování příznaků

Učení konvoluční neuronové sítě je velmi výpočetně náročné. Pokud je učena jen část předučené sítě, zbytek jejích vah je zmražený – nedochází k aktualizaci vah během učení. V tomto případě je výhodné si předpočítat a uložit vektor příznaků jednotlivých snímků, který se stane vstupem do části učené sítě. Velikost vstupního vektoru se může značně lišit podle zvolené architektury sítě. Celý proces zachycuje Obrázek 3.7.

Byly použity state–of–the–art sítě s váhami datové sady ImageNet. Tyto konvoluční neuronové sítě byly naučeny na předzpracovaných snímcích – tedy i naše letecké snímky musí být předzpracovány stejným způsobem.

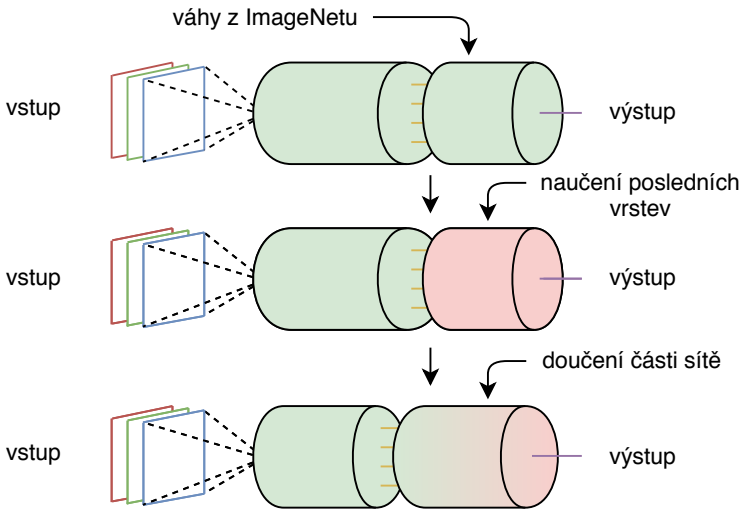



**Obrázek 3.7:** Využití předpočítaných souborů, které ušetří velké množství času při učení sítě.

### 3.5.5 Fáze učení

Jako výchozí model byl zvolen odhad rychlosti vozidel za pomoci maximální povolené rychlosti na daném silničním segmentu a parametrický model 3.5.6. Následně bylo experimentováno s předučenými konvolučními neuronovými sítěmi. První experimenty se týkaly učení posledních plně propojených vrstev. Po naučení těchto vrstev se odmrazila část vah state–of–the–art sítě a proběhlo jejich doučení (*finetuning*). Obě fáze jsou zachyceny na Obrázku 3.8. Všechny fáze používají váhy předučených state–of–the–art sítí na datové sadě ImageNet.

Protože se jedná o regresní úlohu, jako poslední vrstva je použit pouze jeden neuron s aktivační funkcí Leaky ReLU (viz 2.6). Ta je vhodnou aktivační funkcí, protože rychlost vozidla nabývá nezáporných hodnot.




**Obrázek 3.8:** Po načtení vah natrénovaných na datové sadě ImageNet, proběhne postavení několika vlastních plně propojených vrstev pro state–of–the–art konvoluční neuronové sítě. Poté se odmrazí několik posledních vrstev state–of–the–art sítě a připojí se naučená část sítě z prostřední fáze.

### 3.5.6 Parametrický model

Parametrický model byl vytvořen jako výchozí model pro porovnání přesnosti naučené konvoluční neuronové sítě. Jedná se polynomiální regresi – stupeň polynomu $p = 4$ byl experimentálně zvolen. Datová množina je tvořena OSM parametry – počet pruhů, délka silničního segmentu, maximální povolená rychlost a typ silničního úseku. Tyto hodnoty byly také experimentálně vybrány. Pokud hodnoty u silničního segmentu chyběly, byly určeny pomocí heuristik a geometrie viz sekce 2.2.1.1.

### 3.5.7 Učení několika posledních plně propojených vrstev

Předpočítaný vektor příznaků je vstupem do neuronové sítě, která se skládá především z plně propojených vrstev. Množství parametrů závisí jak na počtu vrstev, tak i na počtu neuronů v každé vrstvě. Poslední vrstva obsahuje jen jeden neuron, který predikuje výslednou rychlost. Kromě plně propojených vrstev je použit i dropout kvůli snížení rizika přeučení. Tato nová síť je vložena namísto původních plně propojených vrstev state–of–the–art konvoluční neuronové sítě. U takto postavené sítě je velké množství hyperparametrů, které ovlivňují výslednou přesnost sítě.

### 3.5.8 Finetuning konvolučních neuronových sítí

Načtení vah sítě, která byla původně učena na jiné datové sadě, může přispět ke zvýšení přesnosti sítě viz [48]. Celá myšlenka je postavena na předpokladu, že první vrstvy rozpoznávají především různé hrany, tvary nebo barvy. Tedy tato znalost se dá obecně použít pro velké množství podobných úloh. Vrchní vrstvy jsou poté nahrazeny vlastní naučenou sítí pro konkrétní problém. Podle velikosti datové množiny je poté odmražen určitý počet vrstev, které jsou lehce změněny (doučeny) podle řešeného problému.

Pro tuto fázi učení je důležité zvolit optimalizátor s menším learning rate tak, aby změny vah nebyly příliš velké. To může být dosaženo pomocí optimalizátoru SGD.

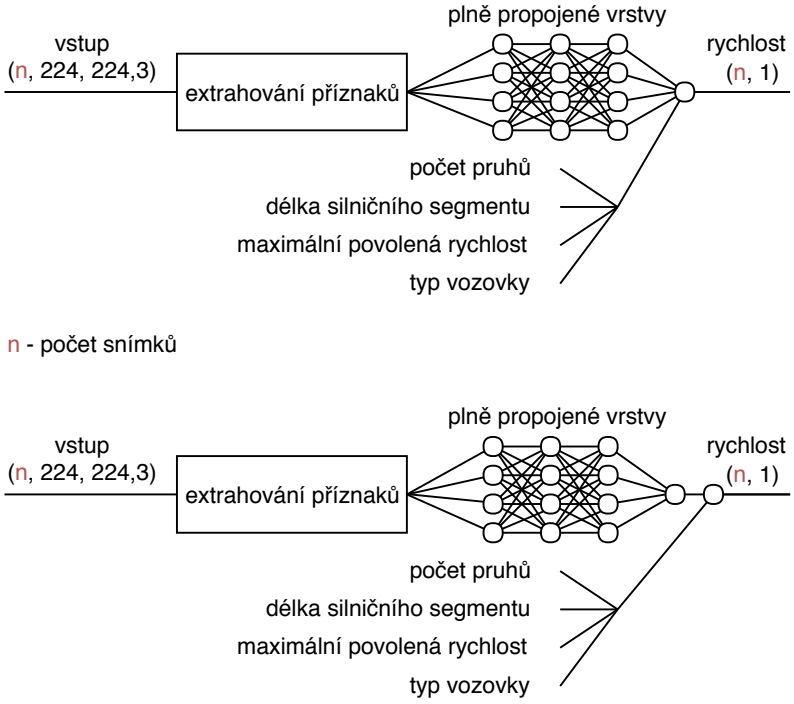
### 3.5.9 Hybridní model

Protože výstupem parametrického modelu je vektor $(1 \times 1)$, je vhodné jeho připojení do konvoluční neuronové sítě až v posledních vrstvách. Pokud by byl připojen níže v síti, jeho vliv na výstup by mohl být velmi malý. Proto byly vytvořeny dva různé typy hybridního modelu zobrazené v Obrázku 3.9.

První typ připojuje výstup z parametrického modelu přímo do poslední plně propojené vrstvy – tedy před neuron, který predikuje rychlost vozidla na silničním segmentu. Druhý typ váží výstupy konvoluční neuronové sítě a parametrického modelu.

## 3.6 Implementace

Popis implementace včetně použitých nástrojů je zmíněn v této kapitole. Práce je členěna do sekcí, které byly implementované zvlášť. Nejprve je ukázaná struktura celého projektu včetně kroků, které jsou nezbytné pro získání predikce rychlosti na silničních segmentech 3.6.1. Poté je popsána tvorba datové množiny včetně augmentace leteckých snímků 3.6.2. Následuje popis knihovny pro optimalizaci hyperparametrů 3.6.3. Celá kapitola je zakončena trénováním modelů 3.6.4 a popisem projektu Metacentra, který byl použit pro náročné výpočty a trénování modelů 3.6.6.




**Obrázek 3.9:** Nahoře je znázorněna konvoluční neuronová síť a parametrický model, jehož výstup je připojen před poslední neuron predikující rychlost vozidla. Dole je znázorněn hybridní model, který pouze váží výstupy (odhadnuté rychlosti) ze sítě a parametrického modelu.

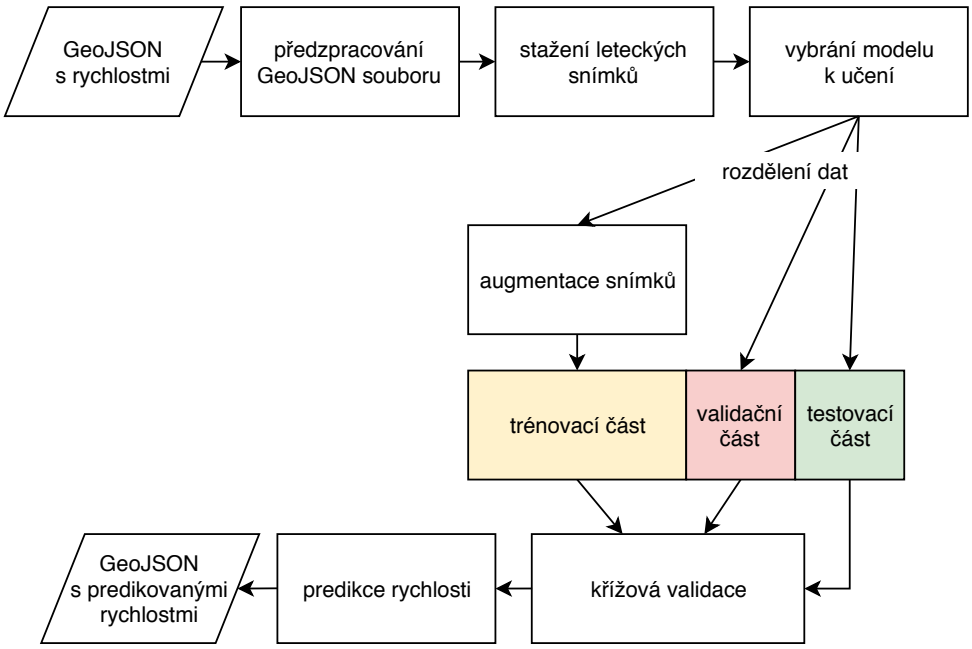
Pro implementaci daného problému byl zvolen programovací jazyk Python[4] ve verzi 3. Hlavní výhodou Pythonu je velké množství dostupných knihoven pro implementaci neuronových sítí. Byla využita knihovna Keras[5] pro modelování neuronových sítí spolu s výpočetní knihovnou TensorFlow[6].

### 3.6.1 Struktura projektu

Nejprve byla implementována funkcionalita, která dokáže stahovat letecké snímky za pomoci Google Maps Static API. Poté byla vytvořena funkcionalita pro načítání, rozdělování a augmentaci snímků. Následně byly vytvořeny modely, jejichž hyperparametry byly experimentálně nastaveny. Vytvořené modely byly učeny podle fází a metod popsaných v sekci 3.5.5. Celý proces potřebný k predikovaní rychlosti za pomoci leteckých snímků je znázorněn na Obrázku 3.10.

Celé nastavení projektu je možné změnit v souboru `config.py`. Nastavení je rozděleno do tří částí – nastavení pro Google Maps Static API, nastavení cest k načítání a ukládání souborů a nastavení konvoluční neuronové sítě.

[4] https://www.python.org
[5] https://keras.io
[6] https://www.tensorflow.org




**Obrázek 3.10:** Schéma znázorňuje proces vzniku výsledného GeoJSON souboru s odhadnutými rychlostmi.

### 3.6.2 Stahování a tvorba dat

Nejdůležitější částí před stahováním snímků je korektní předzpracování vstupního GeoJSON souboru, který obsahuje hrany reprezentující silniční síť. Načtená silniční síť je převedena do grafové struktury za pomoci knihovny NetworkX[7]. Poté pro každou hranu v grafu jsou vytvořeny body, které budou sloužit jako středy stažených leteckých snímků. Každý bod obsahuje informaci s GPS souřadnicemi, skutečnou naměřenou rychlostí a identifikačním číslem hrany, ze které byl vygenerován.

Stažení obrázku je implementováno v `mapdownloader.py`. Pro každý bod je voláno Google Maps Static API. Obdržená data jsou uložena do úložiště a informace o cestě ke snímku je přidána k bodu. Pokud dojde k výpadku internetového připojení nebo bude překročen denní limit Google Maps Static API, budou všechna data dočasně uložena. Poté lze stáhnout pouze chybějící letecké snímky. Průběh stahování dat je zachycen na Obrázku 3.11. Po stažení všech snímků je otestováno, zda jsou všechny ve správném formátu a je možné je číst.

V souboru `data_processing.py` se nachází funkce, které se starají o načítání a extrahování příznaků snímků viz 3.5.4. Pro učení nebyly uvažovány snímky, jejichž skutečná rychlost nebyla v intervalu od 5 $km/h$ do 150 $km/h$. Augmentace dat je implementovaná pomocí knihovny Imgaug[8]. Všechny použité transformace jsou popsány v 3.3.

[7] https://networkx.github.io
[8] https://github.com/aleju/imgaug

```
for point in data['features']:
    coordinates = _get_coordinates(point['geometry']['coordinates'])
    # nastaveni stredu snimku
    settings['center'] = coordinates
    url = STATIC_MAP_BASE_URL + '?' + parse.urlencode(settings)
    response = request.urlopen(url)
    # ulozeni snimku
    save_jpg(filename, response)
    # ulozeni cesty ke snimku
    point['properties']['src'] = path_to_file
```

**Obrázek 3.11:** Ukázka stahování leteckých snímků pomocí Google Maps Static API.

```
# pro hyperparametry modelu zvoli 60 krat nahodne hodnoty a vrati
# nejlepsi nalezene hyperparametry vcetne nauceneho modelu
best_run, best_model = optim.minimize(model=get_model,
                                      data=get_data,
                                      algo=rand.suggest,
                                      rseed=7,
                                      max_evals=60,
                                      trials =Trials())
```

**Obrázek 3.12:** Ukázka optimalizace hyperparametrů konvoluční neuronové sítě za pomoci Hyperas knihovny.

### 3.6.3 Výběr hyperparametrů

Pro optimalizaci hyperparametrů byla použita knihovna Hyperas[9]. Tato knihovna nabízí dva optimalizační algoritmy – TPE, random search. Jak bylo zmíněno v 3.4, random search je doporučený algoritmus pro optimalizaci hyperparametrů. Ukázka volání optimalizace hyperparametrů je zachycena na Obrázku 3.12. Při optimalizaci byla použita trojnásobná křížová validace. Přestože knihovna vrací nejlepší výsledek, lze v proměnné `trials` prohlížet i ostatní výsledky s použitými hodnotami hyperparametrů. Ostatní výsledky mohou být důležité při vybírání optimálního počtu vrstev a počtu neuronů v nich. Zde totiž platí, že je lepší vybrat model, který má menší komplexitu, pokud validační chyba je téměř shodná. Protože většina hyperparametrů nemá lineární vliv na přesnost sítě, byla použita logaritmická stupnice, která umožní prohledat prostor hodnot hyperparametru rychleji [32].

### 3.6.4 Trénování konvolučních neuronových sítí

K učení byla použita knihovna Keras s výpočetní knihovnou TensorFlow. Keras poskytuje jednoduché API, díky němuž lze napsat velmi komplikovanou architekturu neuronové sítě pomocí pár řádek kódu. Keras také podporuje učení na GPU, které sníží dobu potřebnou k učení. Všechny state–of–the–art

[9] https://github.com/maxpumperla/hyperas

```
model = DenseNet121(include_top=False, weights='imagenet',
                    input_shape=(224, 224, 3))

model = ResNet50(include_top=False, weights='imagenet',
                 input_shape=(224, 224, 3))

model = InceptionV3(include_top=False, weights='imagenet',
                    input_shape=(224, 224, 3))

model = VGG16(include_top=False, weights='imagenet',
              input_shape=(224, 224, 3))
```

**Obrázek 3.13:** Načítání state–of–the–art architektur včetně vah natrénovaných na datové sadě ImageNet. Vstupní snímek musí mít rozměry 224 pixelů na výšku i šířku.

sítě jsou v Keras knihovně implementovány včetně možnosti načtení vah, které byly naučeny na datové množině ImageNet. Načtení konvoluční neuronové sítě včetně vah z ImageNet je ukázáno na Obrázku 3.13.

### 3.6.5 Parametrický model

Parametrický model 2.2.1.1 využívá pouze parametry získané z OSM. Pro jeho implementaci byla použita knihovna scikit-learn[10]. Pomocí křížové validace a grid search byly vybrány parametry, které nejvíce vysvětlovaly rozptyl rychlosti a zároveň dosáhly nejmenší validační chyby. Spolu s parametry byl vybrán také stupeň polynomiální regrese.

### 3.6.6 Metacentrum

Na Metacentru byly provedeny veškeré výpočty spojené s trénováním sítí a optimalizací hyperparametrů. Metacentrum je bezplatný projekt pro akademické pracovníky a studenty. Tito uživatelé mají přístup k výpočetní a úložné kapacitě. Metacentrum provozuje řadu výkonných počítačů napříč Českou republikou. Za účelem trénování konvoluční neuronové sítě byly použity stroje osazené grafickými kartami NVIDIA GeForce GTX TITAN X. Tato karta je považována za nejvhodnější grafickou kartu pro hluboké učení [49]. Kromě grafické karty je zapotřebí knihovna cuDNN[11], která efektivně počítá běžné operace v hlubokých neuronových sítích.

Úlohy jsou řazeny do front a jejich spuštění je řízeno systémem PBSPro. Používané příkazy pro vkládání úloh do fronty jsou ukázány na Obrázku 3.14. V rámci řešení této práce bylo propočítáno 378,9 dnů procesorového času, také bylo zapotřebí v součtu okolo 300 GB paměti v úložišti pro uložení všech předpočítaných příznaků snímků.

[10] http://scikit-learn.org/stable/index.html
[11] https://developer.nvidia.com/cudnn

```
# vlozi ulohu task.sh do fronty
qsub -l walltime=4:0:0 -q gpu
-l select =1:ncpus=2:ngpus=1:mem=10gb:gpu_cap=cuda35 task.sh
# maximalni doba behu jsou 4 hodiny
# pridelene jsou 2 procesory na jednom uzlu, 10 GB operacni pameti a
# 1 GPU s CUDA Compute Capability 3.5

# smazani ulohy z fronty
qdel task.sh

# zobrazeni bezicich uloh a uloh ve fronte
qstat -u user_name
```

**Obrázek 3.14:** Příkazy nutné pro vkládání, odstraňování a vypisování úloh v systému PBSPro.

# Kapitola 4
# Výsledky

Tato kapitola je rozdělena do několik částí. Nejprve je popsána tvorba architektury plně propojených vrstev spolu s optimalizací dalších hyperparametrů potřebných pro trénování konvoluční neuronové sítě 4.1. Dále je experimentováno se třemi datovými množinami – snímky pouze ze středu silničního segmentu, snímky z celého silničního segmentu a datová sada vzniklá augmentací snímků z první datové množiny. Poté jsou popsány dosažené výsledky jednotlivých fází učení sítí viz 3.5.5.

Trénování konvoluční neuronové sítě zachycuje na grafech evoluci jak trénovací, tak i validační chyby. Protože k učení byla použita křížová validace 3.5.2, je do každého grafu přidána průměrná chyba pro validační i trénovací množinu. Všechny modely byly učeny 100 epoch, což se ukázalo dostačující vzhledem k tomu, že validační chyba se dále nezlepšovala. Jednou epochou nazýváme předložení celé trénovací množiny konvoluční neuronové síti.

V některých případech, kdy se síť přetrénovala, je žádoucí vybrat váhy z epochy, při které validační chyba byla nejmenší [44]. Níže uvedené experimenty se tímto pravidlem řídí.

## 4.1 Výběr hyperparametrů

Výběr hyperparametrů má signifikantní vliv na rychlost učení a zároveň na výslednou dosaženou přesnost sítě. Nejdůležitější hyperparametr je learning rate [44] – u algoritmu SGD je vybíráno také momentum, které má motivaci z fyzikální perspektivy. Ten byl optimalizován pro tři různé optimalizační algoritmy ztrátové funkce – SGD, RMSProp a Nadam. Výsledné optimální hodnoty jsou zachyceny v Tabulce 4.1.

Mezi dalšími důležitými hyperparametry jsou počet plně propojených vrstev a počet neuronů v každé vrstvě. Nutné je vybrat také vhodnou aktivační funkci. Takto vzniklá architektura je vložena na místo původních plně propojených vrstev state–of–the–art sítě. Výsledné konfigurace sítí uvádí Tabulka 4.2.

V poslední řadě byla optimalizována velikost batche, pravděpodobnost dropoutu v každé vrstvě a přítomnost batch normalizace (Tabulka 4.3).

| **Learning rate** | optimalizátor | | |
|---|---|---|---|
| architektura | SGD | RMSProp | Nadam |
| ResNet50 | lr=0,00044, mom=0,85 | lr=0,00049 | lr=0,00024 |
| VGG16 | lr=0,00001, mom=0,53 | lr=0,000054 | lr=0,00011 |
| Inception | lr=0,01076, mom=0,69 | lr=0,00024 | lr=0,00017 |
| DenseNet121 | lr=0,00060, mom=0,71 | lr=0,00014 | lr=0,00017 |

**Tabulka 4.1:** Tabulka uvádí pro všechny optimalizační algoritmy jejich optimální learning rate (lr) a momentum (mom) pro SGD.

| **Plně propojené vrstvy s aktivační funkcí** | | |
|---|---|---|
| architektura | počet vrstev a neuronů | aktivační funkce |
| ResNet50 | vrstva1=256 | tanh |
| VGG16 | vrstva1=128, vrstva2=256, vrstva3=512 | SELU |
| Inception | vrstva1=128, vrstva2=128 | SELU |
| DenseNet121 | vrstva1=512, vrstva2=256 | tanh |

**Tabulka 4.2:** Tabulka uvádí optimální počet vrstev a počet neuronů v nich. Dále je uvedena zvolená aktivační funkce. U uvedených vrstev pořadové číslo značí vzdálenost od výstupu. Čím je číslo vyšší, tím je vrstva blíže k výstupu.

## 4.2 Předzpracování leteckých snímků

Nejprve bylo experimentálně vybráno předzpracování snímků. Protože byly použity váhy natrénované na datové množině ImageNet, bylo nutné použít stejné předzpracování. Kromě normalizace hodnot pixelů autoři state–of–the–art sítí použili další metody jako je odečtení průměrné hodnoty od každého barevného kanálu nebo změnění pořadí kanálů. Bylo zjištěno, že průměrná hodnota barevných kanálů se liší od průměru získaného na datové sadě ImageNet. Proto bylo testováno, jak ovlivní přesnost předzpracování určené z datové sady leteckých snímků. Ani pro jeden model nebylo pozorováno signifikantní zlepšení viz Obrázek A.2.

Z toho vyplývá, že daleko důležitější pro učení konvoluční neuronové sítě je normalizace hodnot pixelů nebo prohození barevných kanálů. Dále je proto využité výchozí předzpracování Keras, jaké bylo použito na datové množině ImageNet.

## 4.3 Porovnání datových množin

Při řešení daného problému byly použity tři datové sady uvedené v Tabulce 4.4. Nejprve byly učeny pouze vrchních plně propojené vrstvy, které byly nastaveny podle 4.1. Zbylé vrstvy byly zmraženy s přednastavenými váhami z datové sady ImageNet. Průběh učení všech architektur na datové sadě MAX100 je zachycen na Obrázku A.3. Nejlepšího výsledku dosáhla

**Dropout, velikost batche a přítomnost batch normalizace**

| architektura | dropout po vrstvách | batch velikost | batch normalizace |
|---|---|---|---|
| ResNet50 | $0,10\%$ | 32 | ne |
| VGG16 | $0,18\%$, $0,33\%$, $0,43\%$ | 16 | ne |
| Inception | $0,32\%$, $0,16\%$ | 16 | ano |
| DenseNet121 | $0,32\%$, $0,21\%$ | 32 | ne |

**Tabulka 4.3:** Tabulka uvádí pravděpodobnost dropoutu v jednotlivých vrstvách, dále pak batch velikost a přítomnost batch normalizace. Procentuální pravděpodobnost dropoutu je ve stejném pořadí jako vrstvy v Tabulce 4.2.

konvoluční neuronová síť DenseNet121 s průměrnou absolutní chybou na testovací množině $9,11$ $km/h$. K největšímu přeučení došlo u sítě ResNet50 a VGG16. Přeučení také ovlivňuje počet parametrů každé z architektur 4.5. Síť InceptionV3 dosáhla podobnou chybu na trénovací i validační sadě i přesto, že má podobný počet parametrů jako ResNet50.

Na datové množině AUG200, která obsahuje augmentovaná data, ani jedna ze sítí nedosáhla lepšího výsledku než na datové množině MAX100 viz Obrázek A.4. Tento fakt značí, že transformované obrázky nepřidají do modelu více informace.

| datová sada | počet snímků |
|---|---|
| CNTR50 | 50 855 |
| MAX100 | 110 190 |
| AUG200 | 203 420 |

**Tabulka 4.4:** Tabulka uvádí počet leteckých snímků, které obsahuje každá z datových množin.

| architektura | počet parametrů |
|---|---|
| ResNet50 | 24 059 393 |
| VGG16 | 18 091 201 |
| InceptionV3 | 22 047 777 |
| DenseNet121 | 7 610 241 |

**Tabulka 4.5:** Tabulka uvádí počet parametrů použitých architektur konvolučních neuronových sítí, které mohou být měněny během učení.

## 4.4 Finetuning

Odmražení několika posledních předtrénovaných vrstev přináší další zvýšení přesnosti modelu [50]. Finetuning konvolučních neuronových sítí je zachycen na Obrázku A.5. Za účelem finetuningu byla vybrána datová sada AUG200,

která musela být omezena na dvě transformace jednoho snímku. Toto omezení bylo dáno velikostí paměti grafické karty a velikostí extrahovaných příznaků. I přesto sítě VGG16 a InceptionV3 zaznamenaly zlepšení přesnosti v porovnání s oběma datovými sadami (MAX100, AUG200). K nejvýraznějšímu zlepšení pomocí finetuningu došlo u sítě VGG16 o $0,51$ $km/h$. Rozdílné výsledky přesnosti mohou být dosaženy počtem odmražených vrstev, jak popisují autoři [48]. Avšak omezená velikost datové množiny znemožnila odmražení dalších vrstev.

## 4.5 Hybridní model

Poslední experimenty byly provedeny spojením naučené konvoluční neuronové sítě s parametrickým modelem 2.2.1.1. Byly zvoleny dvě strategie propojení těchto modelů. První strategie pouze propojila výstup z konvoluční neuronové sítě s výstupem parametrického modelu. Druhá strategie přidala výstup parametrického modelu do poslední plně propojené vrstvy. Vložení výstupu parametrického modelu hlouběji do sítě nebylo realizováno kvůli počtu neuronů ve vrstvách, které by omezily propagaci informace. Výsledky obou strategií jsou zachyceny na Obrázku A.6. Byla použita datová množina AUG200. U všech architektur lépe dopadla druhá strategie. Jedním z možných vysvětlení je, že v poslední plně propojené vrstvě je více neuronů, z jejichž výstupů lze vybírat. Naproti tomu v první strategii dochází pouze k přidání vah vstupům z konvoluční neuronové sítě a parametrického modelu. Zároveň u všech modelů se jednalo o zlepšení oproti použití pouze konvoluční neuronové sítě, výjimkou byla síť DenseNet121. Možné vysvětlení je takové, že počet parametrů v plně propojené vrstvě je výrazně menší oproti zbylým modelům.

Nejlepší sítě z každé fáze učení a různých datových množin jsou uvedeny v Tabulce 4.6. Kromě průměrné absolutní chyby (MAE) jsou přidány metriky – průměrná druhá mocnina chyby (MSE), medián absolutní chyby (MedAE) a koeficient determinace. Koeficient determinace ($R^2$) je podíl rozptylu závislé proměnné vysvětlený modelem.

Celkově nejlepšího výsledku dosáhl parametrický model spolu s ResNet50 znázorněn modře v Tabulce 4.6. Avšak nejlepší výsledky při učení pouze konvoluční neuronové sítě dosáhla síť DenseNet121 znázorněna červeně v Tabulce 4.6. Jednotlivé vlivy fází a datových množin na síť DenseNet121 jsou zobrazeny na Obrázku 4.1. Distribuce rychlostí hybridního modelu je zachycena na Obrázku A.7.

## 4.6 Vizualizace rychlostí na silniční síti Prahy

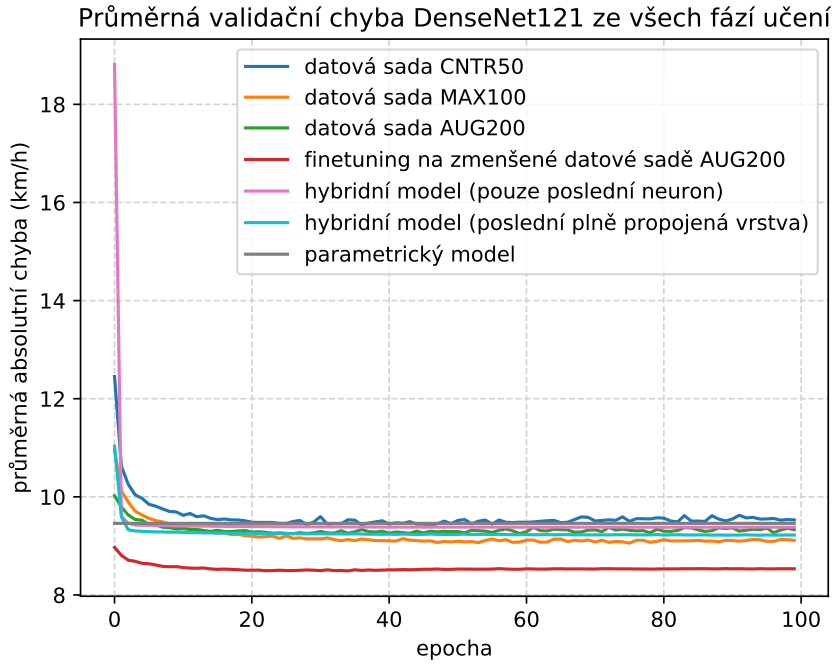
Pro vizualizaci rychlostí byl použit hybridní model (parametrický model, ResNet50), který dosáhl nejmenší chyby na testovacích datech. Protože testovacích dat bylo jen 5 071 náhodně vybraných silničních hran, byla pro vizualizaci použita celá datová sada kromě outlierů.

**Metriky na testovací datové množině**

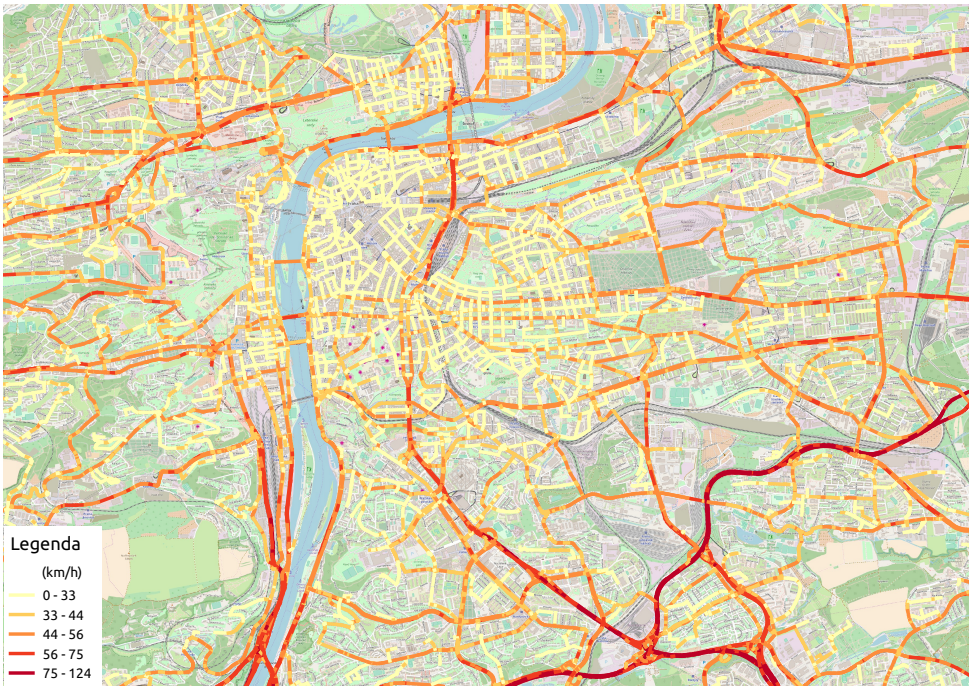
| architektura | MAE | MSE | MedAE | $R^2$ |
|---|---|---|---|---|
| Datová množina CNTR50 | | | | |
| DenseNet121 | $9,71 \pm 0,024$ | $198,58 \pm 0,96$ | $6,95 \pm 0,080$ | $0,47 \pm 0,0026$ |
| Datová množina MAX100 | | | | |
| DenseNet121 | $9,11 \pm 0,025$ | $174,70 \pm 0,90$ | $6,60 \pm 0,054$ | $0,63 \pm 0,0019$ |
| Datová množina AUG200 | | | | |
| DenseNet121 | $9,58 \pm 0,028$ | $193,71 \pm 0,53$ | $6,89 \pm 0,069$ | $0,49 \pm 0,0014$ |
| Finetuning na zmenšené datové množině AUG200 | | | | |
| DenseNet121 | $9,25 \pm 0,022$ | $185,76 \pm 0,60$ | $6,52 \pm 0,043$ | $0,51 \pm 0,0016$ |
| Hybridní model učen na datové množině AUG200 | | | | |
| ResNet50 | $8,61 \pm 0,008$ | $167,67 \pm 0,35$ | $6,16 \pm 0,029$ | $0,56 \pm 0,0009$ |

**Tabulka 4.6:** Tabulka zachycuje nejlepší sítě, které dosáhly nejvyšší přesnosti v dané fázi učení nebo na určité datové sadě. Je uvedená průměrná přesnost na testovací množině včetně směrodatné odchylky.

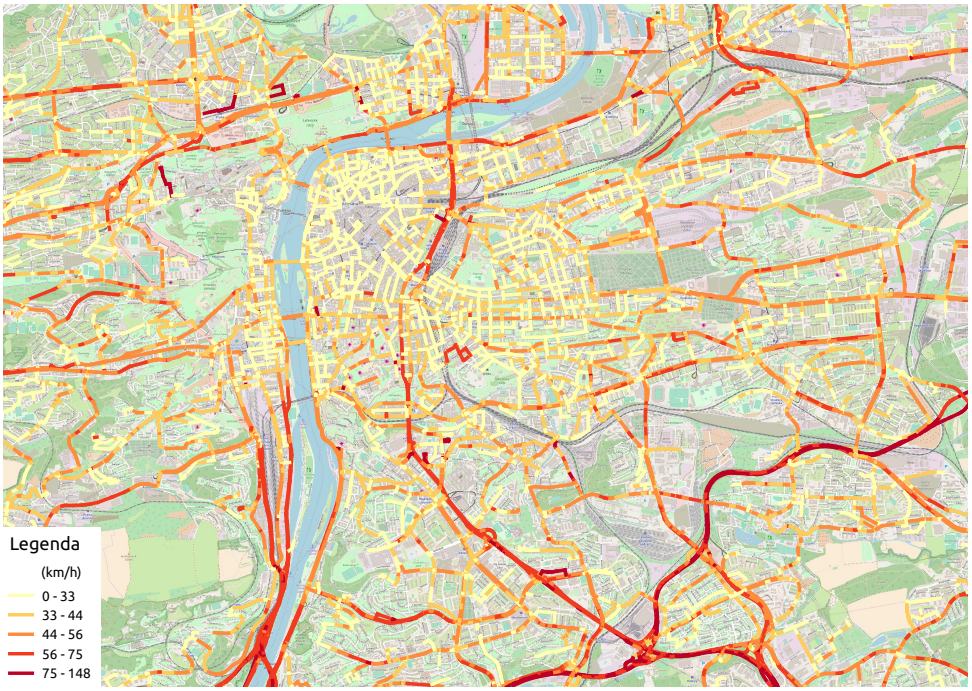
Obrázek 4.2 zachycuje silniční síť Prahy s predikovanými rychlostmi a Obrázek 4.3 zobrazuje skutečné rychlosti na silničních segmentech Prahy. Je možné pozorovat několik outlierů ve skutečných datech, které nebylo možné odstranit. Jedná se o silnice, kde byla naměřena velmi vysoká rychlost v porovnání se sousedními segmenty, avšak nepřekročila limit 150 $km/h$. Je patrné, že rychlost na těchto úsecích přesněji predikuje hybridní model. Absolutní chyba predikce rychlostí hybridního modelu je zachycena na Obrázku A.1. Odhad rychlosti na vybraných leteckých snímcích je ukázán na Obrázku 4.4.



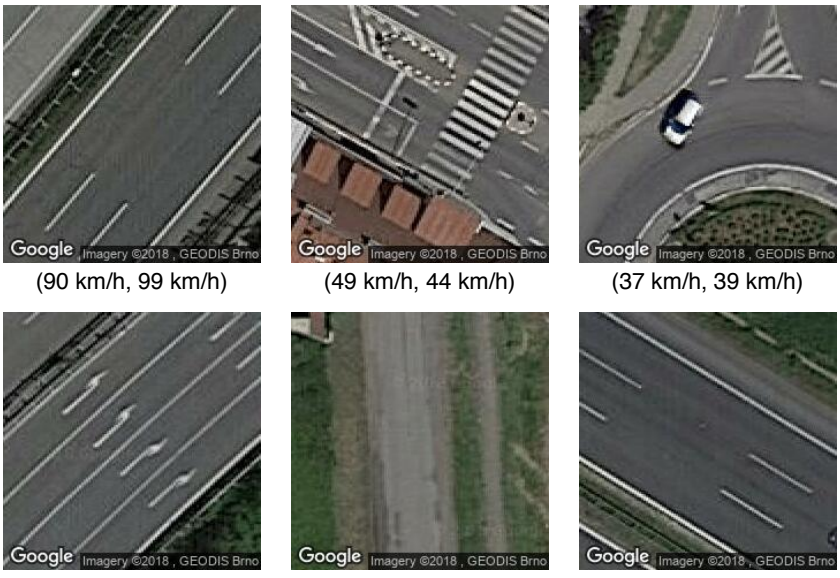

**Obrázek 4.1:** Zobrazení jednotlivých fází učení včetně hybridních modelů na různých datových sadách. Do grafu je pro porovnání přidán také parametrický model.




**Obrázek 4.2:** Predikování rychlosti vozidel na silniční síti Prahy. Pro odhad rychlosti byl použit hybridní model se sítí ResNet50. Kvůli malé testovací množině je k vizualizaci dat použita celá datová množina CNTR50.




**Obrázek 4.3:** Skutečné rychlosti vozidel na silniční síti Prahy.




**Obrázek 4.4:** Příklady leteckých snímků spolu s predikcí rychlosti. U každého snímku je uvedena dvojice (*skutečná rychlost*, *odhadnutá rychlost*).

# Kapitola 5

# Závěr

Predikce rychlosti nachází uplatnění v oblasti plánování optimální trasy a logistiky. Využití lze také nalézt v aplikacích jako je simulace dopravního provozu nebo při navrhování silničních úseků. Současné rychlostní modely využívají především záznamy jízd vozidel nebo vlastností silničních segmentů.

Tato práce popsala důležité techniky používané pro predikci rychlosti vozidel. Byl navržen nový přístup odhadu rychlosti pomocí leteckých snímků silničních úseků. Dostupnost leteckých snímků nám umožnila jejich využití pro odhad rychlosti na silničních úsecích. V úlohách, jejichž datovou množinu tvoří obrázky, je doporučováno použití konvoluční neuronové sítě. U čtyř vybraných state–of–the–art konvolučních neuronových sítí byly použity váhy naučené na datové sadě ImageNet. Navíc byla použita technika uložení vyextrahovaných příznaků snímků, která značně urychlila proces učení. Kromě konvolučních neuronových sítí byl představen hybridní model, který kombinuje výhody z neuronových sítí a parametrických modelů.

Podrobně byl popsán výběr hyperparametrů sítí, které ovlivňují rychlost a přesnost celého učení. Bylo experimentálně ověřeno množství různých způsobů trénování neuronových sítí. Nejlepší přesnosti dosáhl parametrický model spolu se sítí ResNet50, jehož průměrná absolutní chyba byla $8,61 \pm 0,008$ $km/h$. Z konvolučních neuronových sítí nejlépe dopadla architektura DenseNet121 naučená na datové sadě MAX100 s přesností $9,11 \pm 0,025$ $km/h$.

Oproti state–of–the–art rychlostním modelům má odhad rychlosti vozidel pomocí leteckých snímků řadu výhod. Náš model je možný aplikovat na libovolný typ silničního segmentu, oproti parametrickým modelům, které jsou navrhovány pro určitý typ silnic. Zároveň v porovnání s modely využívajícími záznamy jízd vozidel není potřeba žádného sběru dodatečných dat. Pro oblasti, které nemají na většině silničních segmentů chybějící parametry v OSM, je doporučeno použít hybridní model. Naopak pro oblasti, kde chybí velké množství parametrů v OSM, je výhodnější použití modelu, který se skládá jen z konvoluční neuronové sítě.

Další výzkum v této oblasti může zahrnovat jinou velikost nebo zdroj leteckých snímků. Možné je také rozšířit množinu použitých konvolučních neuronových sítí.

# Literatura


[1] TRB Operational Effects of Geometrics Committee et al. Modeling operating speed: synthesis report. *Transportation research circular E-C151*, 2011.

[2] RA Krammes, K Fitzpatrick, JD Blaschke, and DB Fambro. Speed: understanding design, operating, and posted speed. Technical report, 1996.

[3] JL Campbell, CM Richard, JL Brown, MG Lichty, J Graham, and M O'Laughlin. Human factors guidelines for road systems. collection c. *NCHRP Report A*, 600, 2010.

[4] Dieter Pfoser. *Floating Car Data*, pages 321–321. Springer US, Boston, MA, 2008.

[5] Michael Jones, Yanfeng Geng, Daniel Nikovski, and Takahisa Hirata. Predicting link travel times from floating car data. In *Intelligent Transportation Systems-(ITSC), 2013 16th International IEEE Conference on*, pages 1756–1763. IEEE, 2013.

[6] Hongjian Wang, Yu-Hsuan Kuo, Daniel Kifer, and Zhenhui Li. A simple baseline for travel time estimation using large-scale trip data. In *Proceedings of the 24th ACM SIGSPATIAL International Conference on Advances in Geographic Information Systems*, page 61. ACM, 2016.

[7] John Rice and Erik Van Zwet. A simple and effective method for predicting travel times on freeways. *IEEE Transactions on Intelligent Transportation Systems*, 5(3):200–207, 2004.

[8] Lili Huang and Matthew Barth. A novel loglinear model for freeway travel time prediction. In *Intelligent Transportation Systems, 2008. ITSC 2008. 11th International IEEE Conference on*, pages 210–215. IEEE, 2008.

[9] Chun-Hsin Wu, Jan-Ming Ho, and Der-Tsai Lee. Travel-time prediction with support vector regression. *IEEE transactions on intelligent transportation systems*, 5(4):276–281, 2004.

[10] JWC Van Lint. Online learning solutions for freeway travel time prediction. *IEEE Transactions on Intelligent Transportation Systems*, 9(1):38–47, 2008.

[11] Kay Fitzpatrick, Paul Carlson, Marcus Brewer, and Mark Wooldridge. Design factors that affect driver speed on suburban streets. *Transportation Research Record: Journal of the Transportation Research Board*, (1751):18–25, 2001.

[12] Kay Fitzpatrick, Paul J Carlson, Mark D Wooldridge, and Marcus A Brewer. Design factors that affect driver speed on suburban arterials. Technical report, 2000.

[13] Yasser Hassan and Mohamed Sarhan. Modeling operating speed: Synthesis report. chapter 5: Deficiencies in existing speed models. *Transportation Research E-Circular*, (E-C151), 2011.

[14] Maximilian Leodolter, Hannes Koller, and Markus Straub. Estimating travel times from static map attributes. In *Models and Technologies for Intelligent Transportation Systems (MT-ITS), 2015 International Conference on*, pages 121–126. IEEE, 2015.

[15] Jin Xu, Wei Lin, and Yiming Shao. New design method for horizontal alignment of complex mountain highways based on "trajectory–speed" collaborative decision. *Advances in Mechanical Engineering*, 9(4):1687814017695437, 2017.

[16] Filippo Gianmaria Praticò and Marinella Giunta. Modeling operating speed of two lane rural roads. *Procedia-Social and Behavioral Sciences*, 53:664–671, 2012.

[17] Benjamin Penchas, Tobin Bell, and Marco Monteiro. A deep learning approach to vehicle speed estimation. 2017.

[18] Alexey Dosovitskiy, Philipp Fischer, Eddy Ilg, Philip Hausser, Caner Hazirbas, Vladimir Golkov, Patrick van der Smagt, Daniel Cremers, and Thomas Brox. Flownet: Learning optical flow with convolutional networks. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 2758–2766, 2015.

[19] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*, 2014.

[20] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei. ImageNet: A Large-Scale Hierarchical Image Database. In *CVPR09*, 2009.

[21] Xiaolei Ma, Zhuang Dai, Zhengbing He, Jihui Ma, Yong Wang, and Yunpeng Wang. Learning traffic as images: a deep convolutional neural network for large-scale transportation network speed prediction. *Sensors*, 17(4):818, 2017.

[22] Danang Wahyu Wicaksono and Budi Setiyono. Speed estimation on moving vehicle based on digital image processing. *International Journal of Computing Science and Applied Mathematics*, 3(1):21–26, 2017.

[23] Huei-Yung Lin. Vehicle speed detection and identification from a single motion blurred image. In *Application of Computer Vision, 2005. WACV/MOTIONS'05 Volume 1. Seventh IEEE Workshops on*, volume 1, pages 461–467. IEEE, 2005.

[24] Convolutional neural network. `http://ufldl.stanford.edu/tutorial/supervised/ConvolutionalNeuralNetwork/`. Navštíveno: 12.5.2018.

[25] Yann LeCun, Léon Bottou, Yoshua Bengio, and Patrick Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998.

[26] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems*, pages 1097–1105, 2012.

[27] Christian Szegedy, Wei Liu, Yangqing Jia, Pierre Sermanet, Scott Reed, Dragomir Anguelov, Dumitru Erhan, Vincent Vanhoucke, Andrew Rabinovich, et al. Going deeper with convolutions. Cvpr, 2015.

[28] Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. Dropout: A simple way to prevent neural networks from overfitting. *Journal of Machine Learning Research*, 15:1929–1958, 2014.

[29] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016.

[30] Gao Huang, Zhuang Liu, Kilian Q Weinberger, and Laurens van der Maaten. Densely connected convolutional networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, volume 1, page 3, 2017.

[31] Min Lin, Qiang Chen, and Shuicheng Yan. Network in network. *arXiv preprint arXiv:1312.4400*, 2013.

[32] Andrej Karpathy. Cs231n convolutional neural networks for visual recognition. `http://cs231n.github.io`, 2018. Navštíveno: 8.5.2018.

[33] Hui Zou and Trevor Hastie. Regularization and variable selection via the elastic net. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 67(2):301–320, 2005.

[34] Marco Castelluccio, Giovanni Poggi, Carlo Sansone, and Luisa Verdoliva. Land use classification in remote sensing images by convolutional neural networks. *arXiv preprint arXiv:1508.00092*, 2015.

[35] Martin Längkvist, Andrey Kiselev, Marjan Alirezaie, and Amy Loutfi. Classification and segmentation of satellite orthoimagery using convolutional neural networks. *Remote Sensing*, 8(4):329, 2016.

[36] Vladimír Kunc. Deep neural network for satellite image classification using openstreetmap. Master's thesis, Czech Technical University in Prague, 2017.

[37] Yangqing Jia, Evan Shelhamer, Jeff Donahue, Sergey Karayev, Jonathan Long, Ross Girshick, Sergio Guadarrama, and Trevor Darrell. Caffe: Convolutional architecture for fast feature embedding. In *Proceedings of the 22nd ACM international conference on Multimedia*, pages 675–678. ACM, 2014.

[38] Neal Jean, Marshall Burke, Michael Xie, W Matthew Davis, David B Lobell, and Stefano Ermon. Combining satellite imagery and machine learning to predict poverty. *Science*, 353(6301):790–794, 2016.

[39] Kang Liu and Gellert Mattyus. Fast multiclass vehicle detection on aerial images. *IEEE Geoscience and Remote Sensing Letters*, 12(9):1938–1942, 2015.

[40] Volodymyr Mnih and Geoffrey E Hinton. Learning to detect roads in high-resolution aerial images. In *European Conference on Computer Vision*, pages 210–223. Springer, 2010.

[41] Jiuxiang Hu, Anshuman Razdan, John C Femiani, Ming Cui, and Peter Wonka. Road network extraction and intersection detection from aerial images by tracking road footprints. *IEEE Transactions on Geoscience and Remote Sensing*, 45(12):4144–4157, 2007.

[42] Geoffrey E Hinton and Ruslan R Salakhutdinov. Reducing the dimensionality of data with neural networks. *science*, 313(5786):504–507, 2006.

[43] Alexander Jung. imgaug - image augmentation library for machine learning. `https://github.com/aleju/imgaug`, 2017. Navštíveno: 7.5.2018.

[44] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep Learning*. MIT Press, 2016. `http://www.deeplearningbook.org`.

[45] Vít Ružicka. Estimating bicycle route attractivity from image data. Master's thesis, Czech Technical University in Prague, 2017.

[46] James Bergstra and Yoshua Bengio. Random search for hyper-parameter optimization. *Journal of Machine Learning Research*, 13(Feb):281–305, 2012.

[47] Pavel Třasák. Globální optimalizační algoritmy. `http://inggeo.fsv.cvut.cz/wiki/doku.php?id=05_rozbory_presnosti:0508_globalni_optimalizacni_metody`, 2016. Navštíveno: 20.5.2018.

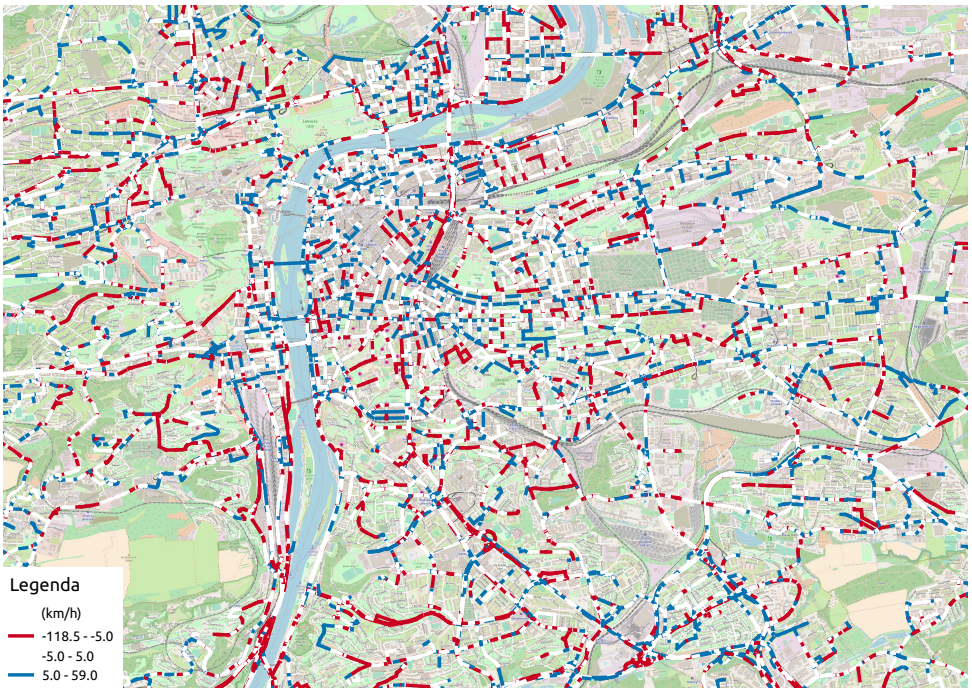
[48] Nima Tajbakhsh, Jae Y Shin, Suryakanth R Gurudu, R Todd Hurst, Christopher B Kendall, Michael B Gotway, and Jianming Liang. Convolutional neural networks for medical image analysis: Full training or fine tuning? *IEEE transactions on medical imaging*, 35(5):1299–1312, 2016.

[49] Kimberly Powell. Nvidia propels deep learning with titan x, new digits training system and devbox. `https://blogs.nvidia.com/blog/2015/03/17/digits-devbox/`, 2015. Navštíveno: 13.5.2018.

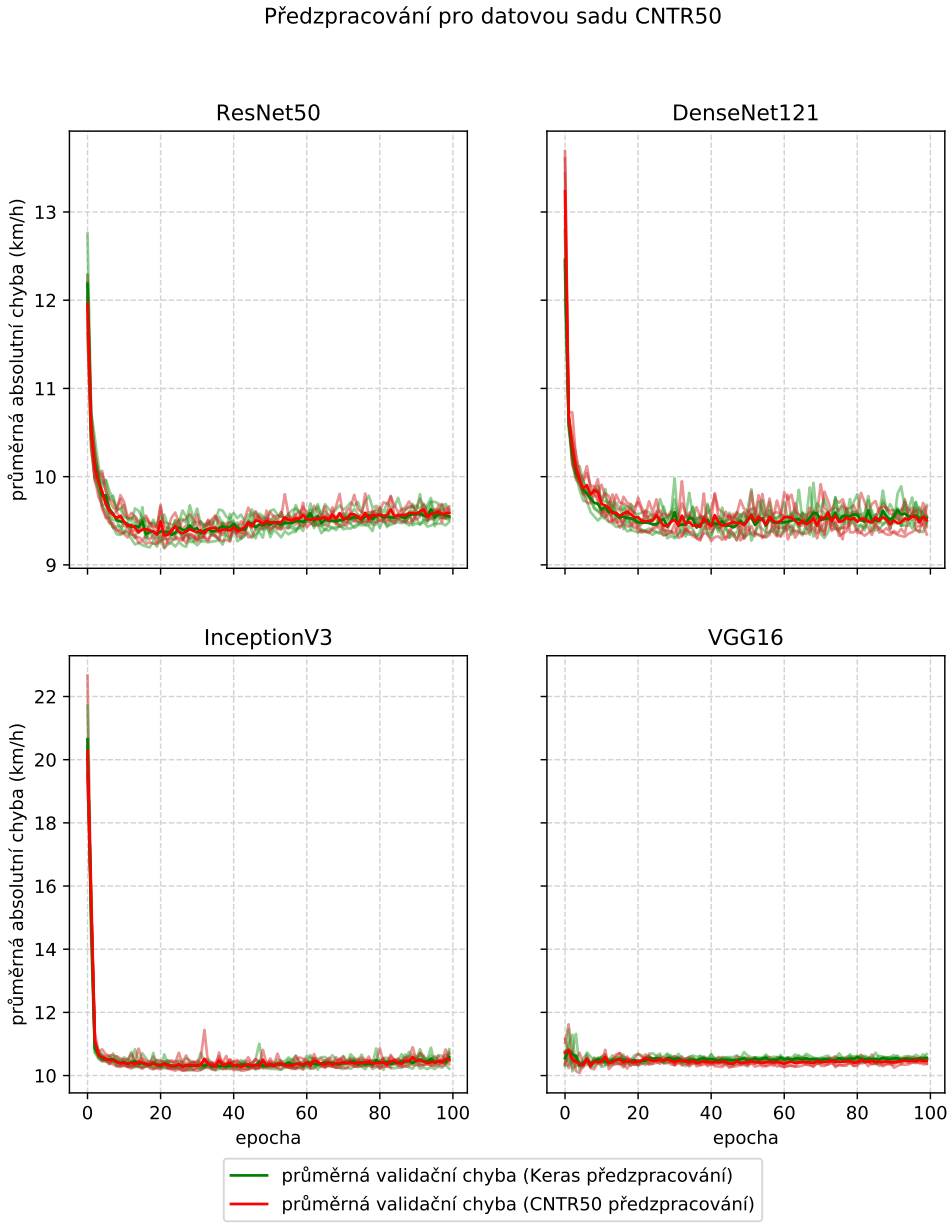
[50] Francois Chollet. Building powerful image classification models using very little data. `https://blog.keras.io/building-powerful-image-classification-models-using-very-little-data.html`, 2016. Navštíveno: 16.5.2018.

# Příloha A

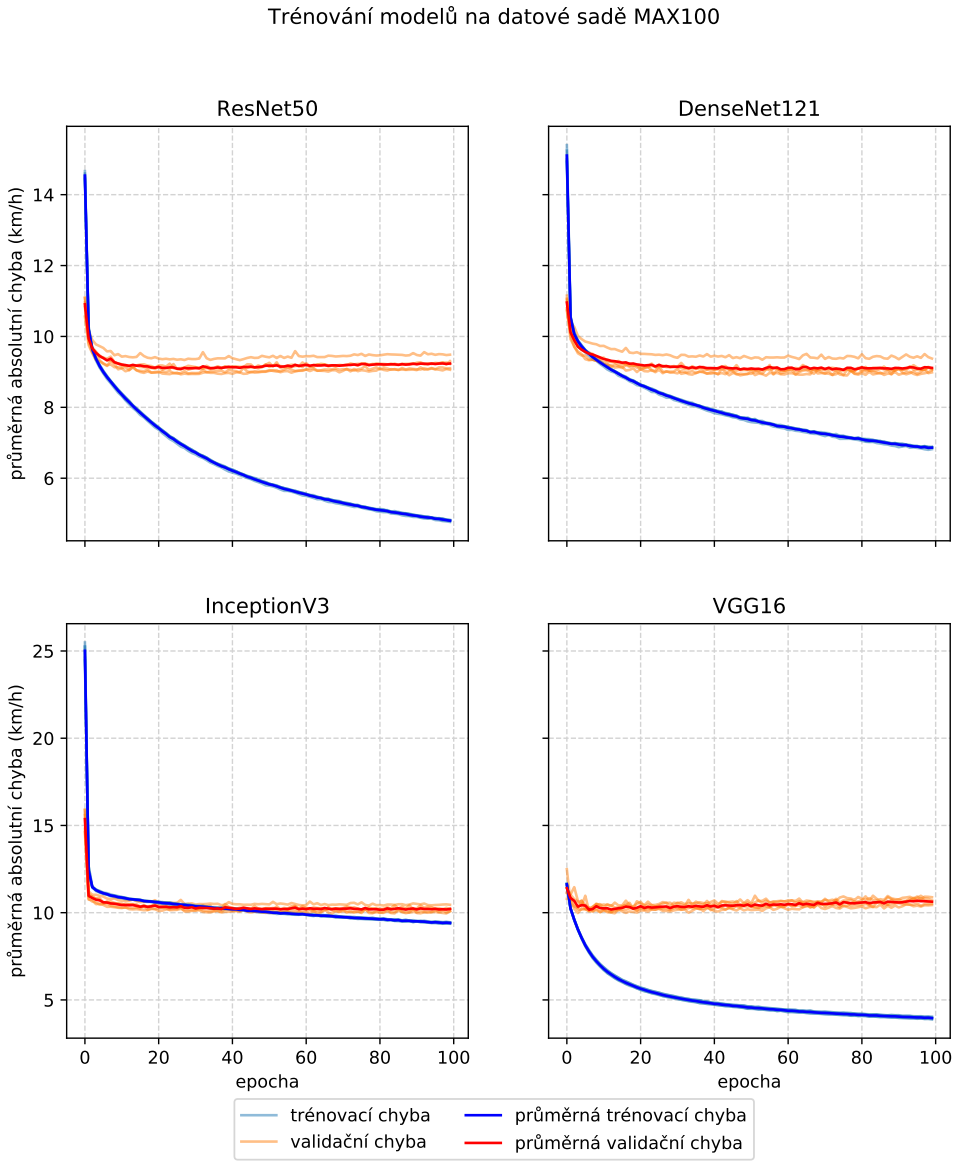
## Další výsledky experimentů a vizualizace



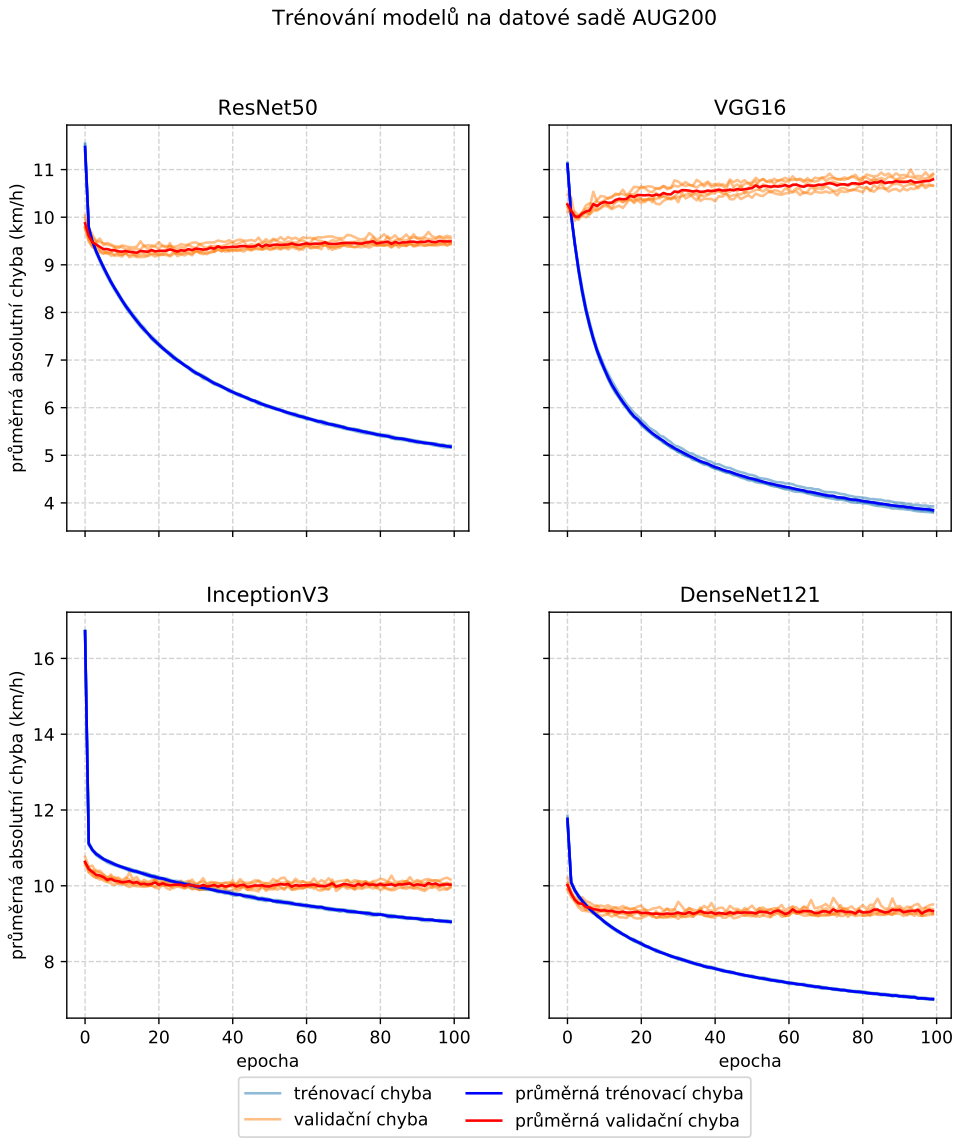

**Obrázek A.1:** Absolutní chyba predikované a skutečné rychlosti na silniční síti Prahy.



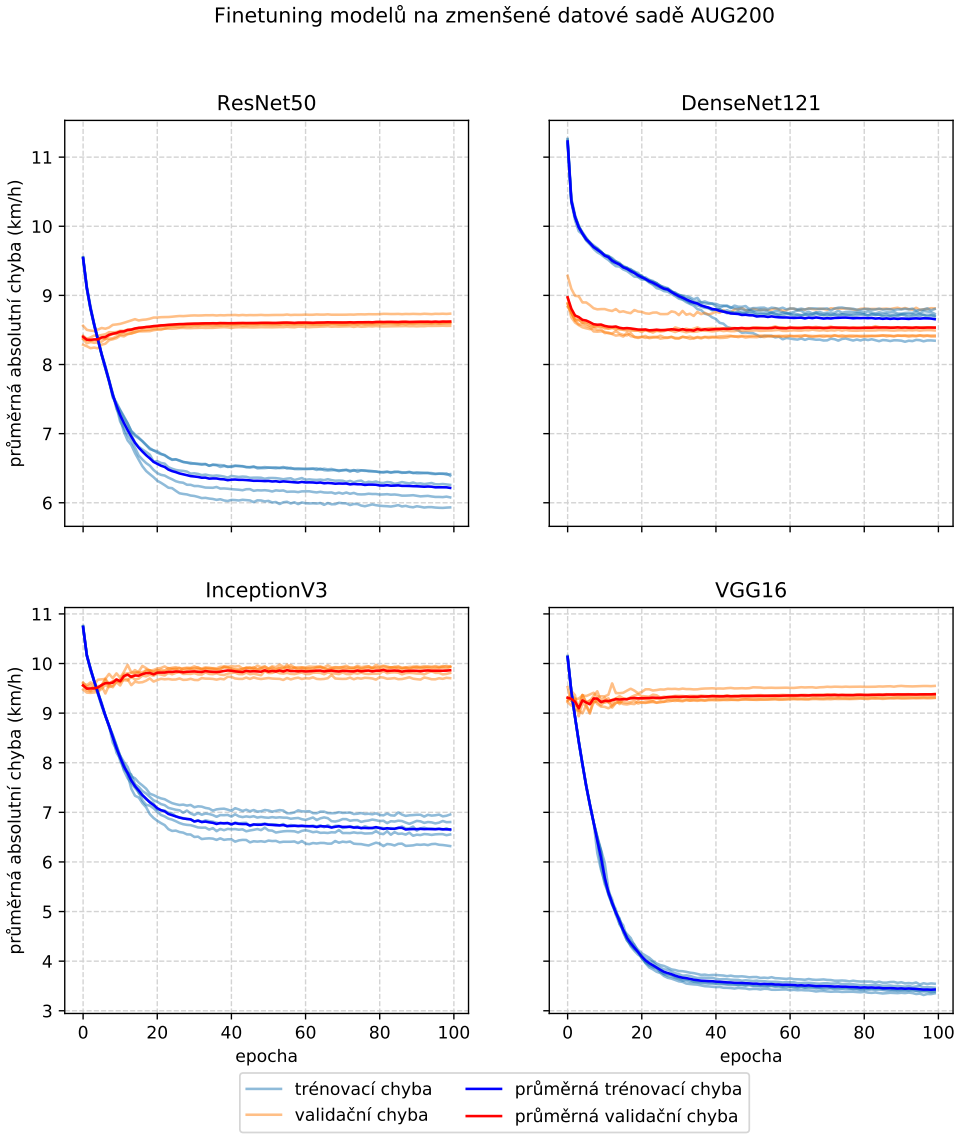

**Obrázek A.2:** Průběh učení konvolučních neuronových sítí s předzpracováním na datové sadě CNTR50 a ImageNet. Pro snazší čitelnost grafů je zobrazena pouze průměrná validační chyba modelů.



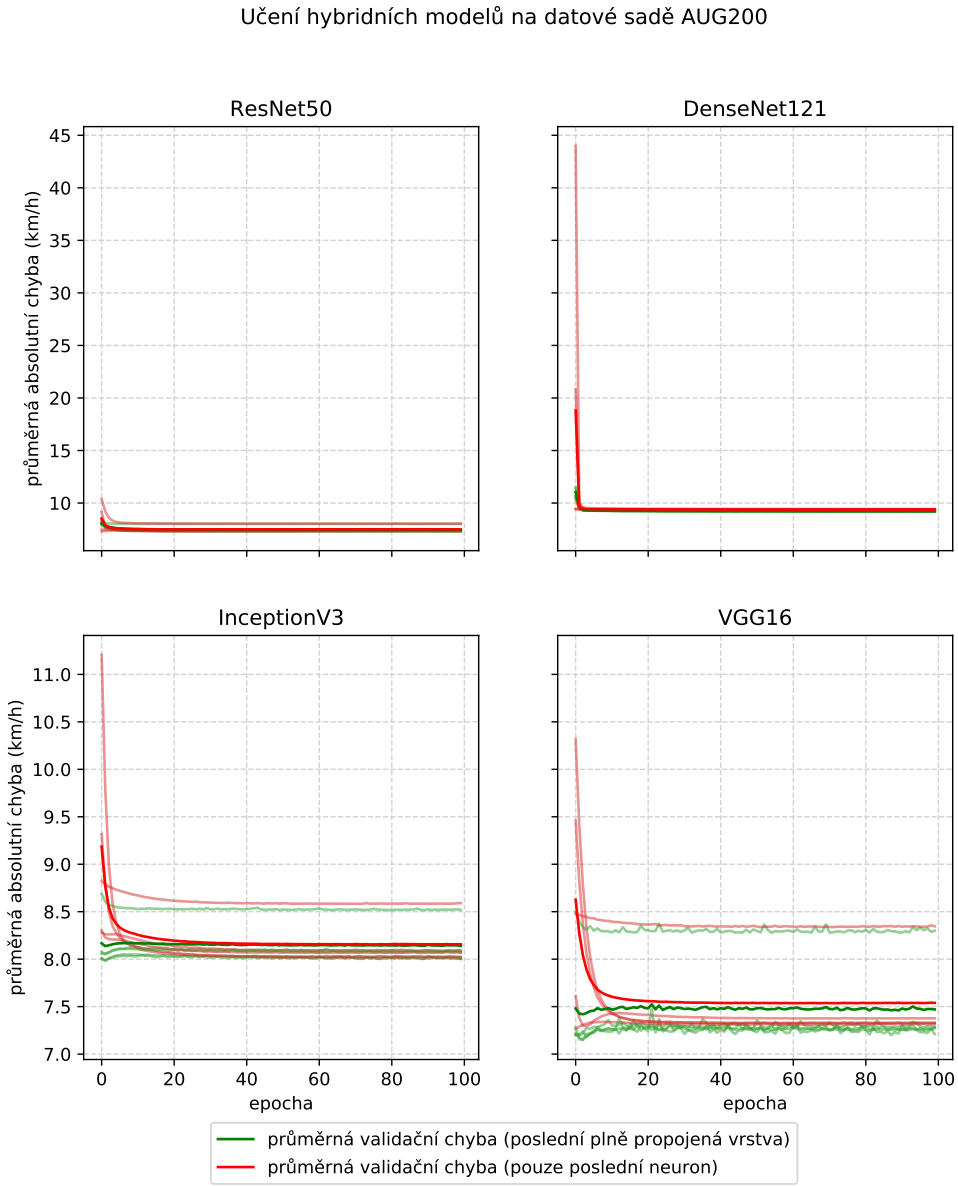

**Obrázek A.3:** Průběh učení konvolučních neuronových sítí na datové sadě MAX100.



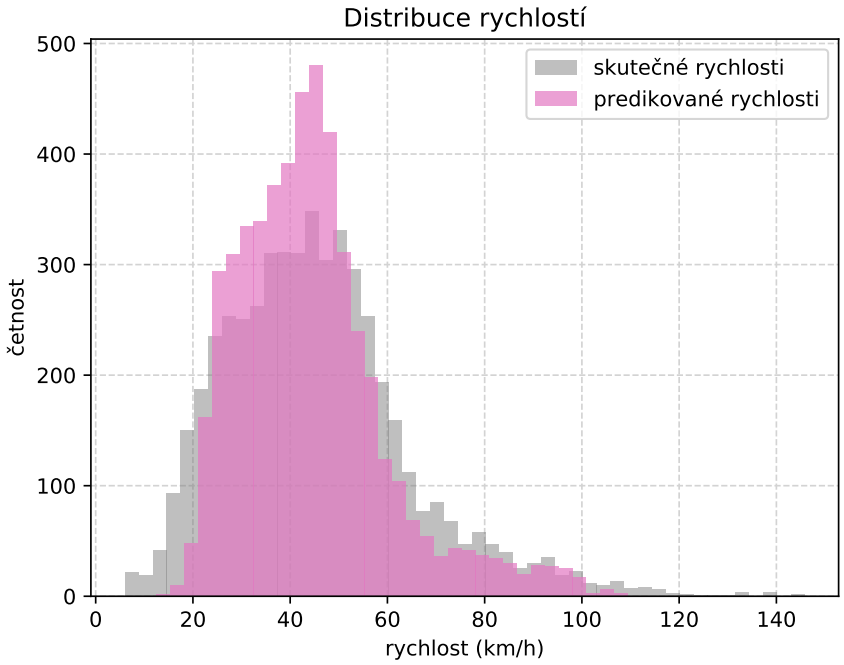

**Obrázek A.4:** Průběh učení konvolučních neuronových sítí na datové sadě AUG200.




**Obrázek A.5:** Finetuning konvolučních neuronových sítí na datové sadě AUG200.




**Obrázek A.6:** Průběh učení hybridních modelů na datové sadě AUG200. Pro snazší čitelnost grafů je zobrazena pouze průměrná validační chyba modelů.




**Obrázek A.7:** Histogram zachycuje predikci rychlostí nejpřesnějšího hybridního modelu se sítí ResNet50. Predikování rychlostí proběhlo na testovací sadě AUG200.

# Příloha B

## Obsah CD

Následující složky jsou nahrány na CD a přiložené k této práci.

- **/Vizualizace** - obsahuje architektury použitých konvolučních neuronových sítí
- **/Výsledky experimentů** - obsahuje všechny experimenty zmíněné v textu spolu s tabulkou všech výsledků a GeoJSON soubory, které obsahují predikované rychlosti
- **/Zdrojový kód** - obsahuje zdrojový kód napsaný v Pythonu spolu se skripty potřebnými ke spuštění experimentů v Metacentru
- **/Práce v LaTeX** - obsahuje tuto práci psanou v LaTeX šabloně a obrázky zde uvedené