Konvolucijska arhitektura za učinkovitu semantičku segmentaciju velikih slika

Krešo, Ivan

University of Zagreb

FACULTY OF ELECTRICAL ENGINEERING AND COMPUTING

Ivan Krešo

CONVOLUTIONAL ARCHITECTURE FOR EFFICIENT SEMANTIC SEGMENTATION OF LARGE IMAGES

DOCTORAL THESIS

Zagreb, 2021

University of Zagreb

FACULTY OF ELECTRICAL ENGINEERING AND COMPUTING

Ivan Krešo

CONVOLUTIONAL ARCHITECTURE FOR EFFICIENT SEMANTIC SEGMENTATION OF LARGE IMAGES

DOCTORAL THESIS

Supervisor: Professor Siniša Šegvić, PhD

Zagreb, 2021

Sveučilište u Zagrebu

FAKULTET ELEKTROTEHNIKE I RAČUNARSTVA

Ivan Krešo

KONVOLUCIJSKA ARHITEKTURA ZA UČINKOVITU SEMANTIČKU SEGMENTACIJU VELIKIH SLIKA

DOKTORSKI RAD

Mentor: Prof. dr. sc. Siniša Šegvić

Zagreb, 2021.

The doctoral thesis was completed at the University of Zagreb Faculty of Electrical Engineering and Computing, Department of Electronics, Microelectronics, Computer and Intelligent Systems.

Supervisor: Professor Siniša Šegvić, PhD

The thesis has 87 pages.

Thesis number: _____

About the Supervisor

Siniša Šegvić was born in 1971 in Split, Croatia. He completed elementary school and high school in Zadar, Croatia, with one year abroad in Milano, Italy. He received the BS degree in electrical engineering (9 semesters) in 1996 as well as the MS and PhD degrees in 2000 and 2004. He has been employed at UniZg-FER as an assistant professor since 2006.

He was a postdoc researcher at IRISA, Rennes and at TU Graz. He led three research projects of the Croatian Science Foundation (MultiCLOD, MASTIF, ADEPT) as well as in several industrial research projects funded by Rimac automobili, RoMB, MicroBlink, and Promet i prostor. He has participated in the research center of excellence DataCross, several ERDF projects (SafeTram, MAS, A-UNIT) as well as on one FP7 project (ACROSS).

His research and professional interests include computer vision, visual recognition, scene understanding, and dense prediction with deep convolutional models. He has published 4 papers at top computer vision conferences (CVPR, ECCV) and 9 papers in journals indexed by WoS. He has been a reviewer at top conferences (CVPR, ECCV, ICCV, AAAI) as well as in scientific journals in the fields of computer vision, intelligent transportation systems and robotics. He participated in the industrial development as a technical consultant. He advises several PhD students funded by EU projects, national projects and private companies. His research group has achieved notable results while participating at computer vision challenges such as WildDash, Robust vision challenge, Cityscapes and Fishyscapes.

Siniša Šegvić speaks english and italian very well, and has basic communication skills in french. He is married and has three children. He is a member of IEEE.

O mentoru

Siniša Šegvić rođen je 1971. u Splitu. Osnovnu školu i gimnaziju završio je u Zadru osim osmog razreda osnovne škole kojeg je pohađao u Milanu. Diplomirao je elektrotehniku na zagrebačkom ETF-u (1996), a tamo je i magistrirao (2000.) i doktorirao (2004.) te se zaposlio kao docent od 2006. godine.

Bio je postdoktorski istraživač na institutu IRISA u Rennesu (2006.-2007.) te na TU Graz (2007.-2008.). Vodio je tri istraživačka projekta Hrvatske zaklade za znanost (MultiCLOD, MASTIF, ADEPT) te više industrijskih istraživačkih projekata koje su financirale tvrtke Rimac automobili, RoMB, MicroBlink te Promet i prostor. Sudjelovao je u istraživačkom centru izvrsnosti DataCross, na nekoliko ERDF projekata (SafeTram,

MAS, A-UNIT) kao i na jednom projektu iz programa FP7 (ACROSS).

Njegovi istraživački i profesionalni interesi uključuju računalni vid, strojno učenje, razumijevanje scena, i gustu predikciju dubokim konvolucijskim modelima. Objavio je 4 rada na vrhunskim konferencijama računalnog vida (CVPR, ECCV) te 9 radova u časopisima koje indeksira SCI. Recenzent je na vrhunskim konferencijama računalnog vida i umjetne inteligencije (CVPR, ECCV, ICCV, AAAI) kao i u znanstvenim časopisima u područjima računalnog vida, inteligentnih transportnih sustava i robotike. Sudjelovao je u industrijskom razvoju kao tehnički konzultant. Mentorira više doktoranada koje financiraju evropski projekti, nacionalni projekti i privatne tvrtke. Njegova istraživačka grupa postigla je zapažene rezultate na nekoliko natjecanja u računalnom vidu (WildDash, Robust vision challenge, Cityscapes i Fishyscapes).

Siniša Šegvić odlično govori engleski i talijanski, i ima osnovne komunikacijske vještine na francuskom. Oženjen je i ima troje djece. Član je IEEE.

Acknowledgements

First, I would like to thank my advisor Siniša Šegvić. His passion, support and encouragement was essential and motivated me to pursue new ideas and accomplish the results presented in this thesis. I have been fortunate to learn from Siniša during these years.

I would also like to thank Josip Krapac who gave me great advice while I was working on the research ideas and writing papers.

Furthermore, I would like to thank other members of the computer vision research group at the faculty who with interesting discussions and collaboration created an encouraging and positive work environment.

Last but not least, a big thank you to my girlfriend, sister, parents and friends who believed in me and gave me support and motivation during this PhD research.

Abstract

This thesis investigates the semantic segmentation of large natural images. We focus on the type of images that are recorded with a camera mounted on a vehicle. These kinds of images do not suffer from the photographer bias and therefore usually contain harder examples to generalize to. For the task of semantic segmentation, this means that the objects appear on a wide range of scales. Hence, it is important that the method works well both for small objects further away and large objects near the camera. The focus of the thesis is on applying convolutional neural networks for semantic segmentation of large images in an efficient manner. The thesis starts by providing an introduction to the problem of semantic segmentation and explaining its relation with respect to the problem of object localization. The introduction is concluded by discussing the challenges of the problem. The next chapter starts by introducing the required concepts from the field of machine learning. In particular, we review the convolutional neural networks and make a comparison between DenseNet and ResNet architecture. The main contributions of the thesis are as follows. First, we develop a scale-invariant convolutional model for semantic segmentation which alleviates the problem of learning the same object on a wide range of scales. Furthermore, we additionally contribute a new dataset for semantic segmentation of driving scenes. The dataset contains groundtruth semantic segmentations for 445 annotated hand-picked images from the KITTI dataset. Second, we develop an efficient asymmetric architecture for dense prediction on large images based on a densely connected feature extractor and lightweight ladder-style upsampling. Third, we present the results from the Robust Vision Challenge 2018 where we achieved the second place. The challenge addressed cross-dataset and cross-domain training of dense prediction models. We have found out that mixed batches ensure the most stable evolution of batchnorm parameters. Low incidence of foreign classes indicates that our models succeeded to implicitly learn to distinguish the domains. Finally, the presented architecture is computationally and memory efficient and achieves a great tradeoff between inference speed and generalization accuracy.

Keywords: computer vision, deep learning, semantic segmentation, ladder networks, convolutional neural networks, Cityscapes, Pascal VOC2012, CamVid, KITTI

Konvolucijska arhitektura za učinkovitu semantičku segmentaciju velikih slika

Ova doktorska disertacija proučava semantičku segmentaciju velikih prirodnih slika. Fokus disertacije je primjena dubokih konvolucijskih modela odnosno konvolucijskih neuronskih mreža za problem semantičke segmentacije. Glavni doprinosi disertacije su sljedeći:

1. učinkovita asimetrična arhitektura za semantičku segmentaciju koja se sastoji od složenog gusto povezanog modula za raspoznavanje te jednostavnih modula za piramidalno sažimanje te ljestvičasto naduzorkovanje;
2. modul za izlučivanje invarijantnih konvolucijskih reprezentacija utemeljen na gustom odabiru mjerila s obzirom na lokalnu stereoskopsku dubinu;
3. metodologija za učenje semantičke segmentacije na višedomenskim podatcima.

Disertacija je organizirana u šest poglavlja. U nastavku su dani sažeci svakog poglavlja.

Prvo poglavlje: Uvod

Mnogi bi se složili da je vid najvažnije osjetilo kojim ljudi percipiraju svijet. Za kretanje primarno koristimo vid. Vid koristimo svakodnevno kako bismo riješili raznovrsne probleme. I naši snovi su vizualni. Stoga nije čudno da ako želimo naučiti računala da obavljaju velik broj naših poslova, najprije ih moramo naučiti da vide. Upravo je cilj područja računalnog vida omogućiti računalima dar vida. Primjene računalnog vida na konkretne probleme su brojne. Neke od njih su: autonomna navigacija robota i vozila, analiza medicinskih snimki, sigurnosni i nadzorni sustavi, biometrijska identifikacija, pretraživanje slika na temelju sadržaja, obrada videa i fotografija, rekonstrukcija scene za 3D karte ili kulturne znamenitosti itd. Koliko god nam se činilo da je vid lagan i prirodan za nas ljude, računalni vid je jedan od najtežih problema unutar područja računarske znanosti.

Semantička segmentacija je važan zadatak računalnog vida s mnogo primjena u robotici, inteligentnom transportu, medicini, obradi slike itd. Zadatak semantičke segmentacije je klasificirati piksele slike u semantičke razrede poput osobe, automobila, drveća, ceste, neba itd. Razred piksela određen je vrstom objekta ili stvari koji se projicira na njega. Ulaz u model je slika, a izlaz je predikcija razreda na razini piksela. Ovaj problem povezan je s poznatijim problemom detekcije objekata koja se bavi pronalaskom pravokutnih okvira objekata u slici. Okvir objekta je minimalni pravokutnik koji sadrži objekt te definira njegov raspon. Broj pronađenih okvira odgovara broju objekata u slici. Ovo je razlika u odnosu na semantičku segmentaciju kod koje broj objekata prilikom učenja i predikcije nije poznat. Upravo ovo je jedan of glavnih izazova semantičke segmentacije: model mora

naučiti prepoznati piksel na temelju odgovarajuće velikog prostornog konteksta. Pri tome veličina konteksta jako varira ovisno o vrsti i veličini objekta u slici. Veličinu objekta u slici dodatno ovisi o vrsti objekta i udaljenosti od kamere. Na primjer za kamion ćemo morati uzeti veći kontekst u obzir nego za automobil ili bicikl. Isto tako za automobil u blizini u odnosu na automobil u daljini.

Konvolucijske mreže prvotno su primijenjene za problem klasifikacije slike. Kod klasifikacije slike cilj je prepoznati da se objekt ili stvar pojavljuju u slici, bez informacije o lokaciji samog objekta. Konvolucijske neuronske mreže vrlo su prikladan model i za problem semantičke segmentacije budući da zbog sažimanja mogu imati jako veliko receptivno polje i tako pokriti velik raspon prostornog konteksta. Međutim, prostorno sažimanje dovodi do smanjenja rezolucije dubokih latentnih reprezentacija. Izgubljenu rezoluciju potrebno je vratiti kako bi predikcija sadržavala manje objekte i točne granice prijelaza između instanci objekata.

U narednim poglavljima, disertacija predlaže efikasna rješenja navedenih izazova. Posebna je pažnja usmjerena na primjenu u inteligentnim robotičkim i prometnim sustavima gdje je kamera postavljena na vozilo ili robota. Slike pribavljene na ovakav način nisu pristrane poput fotografija predmeta koju su snimili ljudi. To dodatno otežava generalizaciju modela učenih na takvim slikama.

Drugo poglavlje: Duboki konvolucijski modeli

Konvolucijske neuronske mreže posebno su prikladne za primjene u računalnom vidu. Razlog tome je njihova prilagođenost topološki organiziranim podatcima gdje je pogodno ostvariti invarijantnost na operaciju translacije. Ovakva pretpostavka dobro odgovara slikama gdje se isti objekt može pojaviti na bilo kojoj lokaciji unutar slike. Konvolucijske mreže u osnovnom obliku sastoje se od slijeda operacija konvolucije i nelinearnosti koji se ponavljaju. Konvolucijski sloj sastoji se od grupe filtara. Svaki od njih provodi operaciju konvolucije nad ulazom i generira mapu značajki na izlazu. Ulazni tenzor može imati dvije, tri ili četiri dimenzije, ovisno o prirodi podataka. Filtri su definirani težinama koji se tijekom učenja ugađaju kako bi imali visok odziv na podatkovnim uzorcima koji su korisni za rješavanje problema. Izlazni tenzor sadrži konvolucijske značajke ili aktivacije. Rani plitki slojevi uče značajke niske razine poput rubova, boje i teksture, a kasniji dublji slojevi uče značajke visoke razine poput očiju, usta, glave, tijela itd. Pored konvolucijskih i slojeva nelinearnosti obično imamo još i slojeve sažimanja i slojeve normalizacije. Uloga slojeva sažimanja je povećanje receptivnog polja modela i ujedno smanjenje prostorne dimenzionalnosti podataka. Slojevi normalizacije omogućuju stabilniju i bržu optimizaciju modela.

Konvolucijske mreže prvotno su razvijene za problem klasifikacije na razini slike. To

ih čini neprikladnima za izravnu primjenu u semantičkoj segmentaciji. Razlog tome leži u sažimanju koje znatno povećava prostorni kontekst i također znatno smanjuje rezoluciju predikcije. Na primjer, modeli prilagođeni za skup podataka ImageNet na izlazu konvolucijskog dijela modela tipično daju 32 puta manju visinu i širinu slike. Izgubljenu rezoluciju potrebno je vratiti kako bi semantička segmentacija imala gustu i detaljnu predikciju s točnim granicama objekata. Ovo nije jednostavno postići zbog memorijskih ograničenja današnjih grafičkih kartica i ostalih akceleriranih čipova. Poglavlje daje pregled pristupa adaptiranja konvolucijskih mreža namijenjenih za klasifikaciju slika za dobivanje guste predikcije potrebne semantičkoj segmentaciji.

Najpopularniji pristup zasniva se na dilatiranim konvolucijama koje ostvaruju veliko receptivno polje bez upotrebe operacija sažimanja. Osnovna ideja je rijetko uzorkovanje ulaznog tenzora gdje se prilikom računanja odziva preskače određeni broj vrijednosti ulaznog tenzora. Veličina rupe između susjednih lokacija uzorkovanja ulaznog tenzora naziva se i dilatacijski faktor. Pomoću ovog faktora možemo jednostavno namjestiti željeno receptivno polje. Kod adaptiranja klasične konvolucijske arhitekture za klasifikaciju najprije se izbacuje operacija sažimanja, a zatim se dilatacijski faktor svih naknadnih konvolucija množi s veličinom koraka iz uklonjene operacije sažimanja. Prednost dilatiranog pristupa je da nisu potrebni dodati slojevi za dobivanje guste predikcije. Mana je znatno povećavanje vremenske i prostorne složenosti modela.

Naš pristup zasniva se na vraćanju rezolucije putem asimetričnog ljestvičasto povezanog modela. Osnovna ideja je miješanje reprezentacija na različitim rezolucijama iz odgovarajućih dijelova mreže. Preciznije, putem preskočne veze miješaju se dublje značajke niže rezolucije s ranijim značajkama na višoj rezoluciji. Model se tako dijeli na dva puta: put poduzorkovanja čiji zadatak je raspoznavanje te put naduzorkovanja čiji zadatak je miješanje reprezentacija različitih rezolucija kako bi se ostvarila gusta predikcija. Dublje značajke su semantički bogate, ali imaju nisku prostornu rezoluciju izgubljenu prilikom sažimanja. Značajke ranijih slojeva bogate su informacijom o lokaciji, ali isto tako sposobne naučiti semantiku manjih objekata i razreda koji ne zahtijevaju veliko receptivno polje. Za razliku od dilatiranog modela, ljestvičasto povezani model ne zahtijeva nepotrebnu propagaciju značajki kroz sve slojeve mreže. Ako je objekt bio prepoznat u ranijim slojevima njegova reprezentacija se može preskočnom vezom direktno proslijediti u put naduzorkovanja.

Treće poglavlje: Konvolucijska invarijantnost na mjerilo za semantičku segmentaciju

U ovom poglavlju prikazan je rad objavljen na konferenciji GCPR 2016 [1]. Jedan od osnovnih problema raspoznavanja objekata u slici je širok raspon mjerila na kojem se objekti pojavljuju. To je posebno izraženo kod prirodnih slika, na primjer slike pribavljene kamerom montiranom na vozilo za vrijeme vožnje.

Osnovna ideja u ovom poglavlju je konvolucijski modul za postizanje invarijantnosti na mjerilo odnosno veličinu objekata. Invarijantnost je ostvarena pomoću slikovne piramide i dubinske slike pribavljene iz stereo kamere. Slikovna piramida omogućuje nam dobivanje ekvivarijantne reprezentacije, dok invarijantnost postižemo normalizacijom mjerila. Dubinska slika omogućuje nam da normaliziramo veličinu na kojoj konvolucijska mreža vidi objekte. Svaki piksel invarijantne reprezentacije odabire se iz odgovarajuće razine slikovne piramide ovisno o vrijednosti dubine u tom pikselu.

Ovdje je također prikazan i skup podataka kojeg smo ostvarili ručnim anotiranjem (označavanjem) podskupa slika iz skupa KITTI. Počeli smo od 146 slika koje su anotirali Ros et. al. [2], poboljšali točnost anotacija i popravili greške. Zatim smo anotirali dodatnih 299 slika. Anotacije su opisane poligonima visoke kvalitete. Konačan skup sastoji se od 445 anotiranih slika.

Za ekstrakciju značajki korištena je VGG-16 konvolucijska mreža. Dubinska slika izračunata je pomoću naučene duboke metrike. Konvolucijski model za duboku metriku naučen je na KITTI stereoskopskom skupu slika gdje su podaci snimljeni LIDAR-om korišteni kao točna dubina. Ulaz u model je slikovna piramida s 8 razina. Najprije se za svaku sliku nezavisno izračunaju konvolucijske značajke. Dobivene značajke zajedno s dubinskom slikom su ulaz u modul za selekciju mjerila. Zadatak modula za selekciju mjerila je predstaviti svaki piksel slike s konvolucijskim značajkama invarijantnim na skalu koje su izračunate na 3 od 8 razina piramide.

Rezultati na skupovima Cityscapes i KITTI pokazuju da ovakav pristup može povećati točnost u odnosu na različite varijante jednostavnijih pristupa koji nisu invarijantni na skalu.

Četvrto poglavlje: Efikasni ljestvičasto povezani DenseNet za semantičku segmentaciju velikih slika

U ovom poglavlju prikazan su radovi objavljeni u časopisu IEEE T-ITS [3] te na radionici CVRSUAD u sklopu konferencije ICCV 2017 [4]. Predložili smo pristup za dobivanje gustih predikcija koji se zasniva na miješanju reprezentacija na različitim rezolucijama iz različitih dijelova mreže. Miješanje reprezentacija izvedeno je pomoću preskočnih veza

od ekstraktora značajki prema modulima za naduzorkovanje. Za razliku od prethodne arhitekture UNet, naša arhitektura je asimetrična: put za naduzorkovanje ima puno manje slojeva od ekstraktora značajki. Ekstraktor značajki izveden je kao klasična konvolucijska arhitektura koja sažimanjem znatno smanjuje prostorne dimenzije latentnih značajki u odnosu na ulaznu rezoluciju. Ekstraktor značajki čuva najapstraktnije latentne reprezentacije na svakoj rezoluciji. Zatim se u povratnom putu dizanja rezolucije ili naduzorkovanja postepeno miješaju značajke na manjoj rezoluciji sa značajkama veće rezolucije koje se preskočnim vezama dovode iz ekstraktora značajki.

Put naduzorkovanja sastoji se od nekoliko jednostavnih i identičnih modula. Ulaz u modul za dizanje rezolucije su značajke na dvije rezolucije. Nad značajkama niže rezolucije najprije se provodi naduzorkovanje bilinearnom interpolacijom. Nad značajkama više rezolucije aplicira se 1×1 konvolucija tako da se broj mapa značajki izjednači reprezentaciji niže rezolucije. Zatim se dobivene reprezentacije miješaju zbrajanjem te se rezultat obrađuje s 1×1 i 3×3 konvolucijama.

Značajke iz različitih dijelova konvolucijskog modela imaju različit omjer lokacijske točnosti i veličine konteksta. Ovakav pristup omogućuje točnu klasifikaciju piksela koji pripadaju i velikim i malim objektima, jer veliki objekti trebaju veliki prostorni kontekst dok maleni objekti trebaju veliku rezoluciju. U odnosu na prethodne pristupe za dobivanje guste reprezentacije klasifikacijskih modela, naš dizajn ima vrlo efikasno progušćivanje predikcije.

Kao bazni model za raspoznavanje odabran je DenseNet-121 te je predložena strategija memorijski efikasnog pristupa učenja ovog modela. Modelu je dodano dodatno sažimanje na sredini trećeg bloka kako bi se dodatno povećalo receptivno polje za velike ulazne slike. U putu za raspoznavanje najprije se reprezentacija sažima 64 puta, a zatim se pomoću četiri efikasna modula za podizanje rezolucije vraća na 4 puta sažetu reprezentaciju. Konačno, bilinearnom interpolacijom dobiva se predikcija na punoj rezoluciji.

Eksperimentalna analiza potvrđuje prednosti opisanog pristupa. Rezultati na skupovima podataka Citscapes, CamVid, ROB2018 te Pascal VOC2012 pokazuju odličan omjer između efikasnosti modela te segmentacijske točnosti.

Peto poglavlje: Robust Vision Challenge

U ovom poglavlju prikazani su rezultati dobiveni na natjecanju Robust Vision Challenge 2018. Cilj natjecanja bio je izmjeriti robusnost metoda računalnog vida na različite domene podataka. Naš pristup natjecao se u problemu semantičke segmentacije te osvojio drugo mjesto. Problem je uključivao četiri različita skupa podataka anotiranih slika iz dvije domene: vanjske snimke vožnje pribavljene automobilom te unutarnje snimke životnog prostora pribavljene fotoaparatima u kućama i stanovima. Prva tri skupa su

sadržavala slike vožnje: Cityscapes, WildDash i KITTI. Zadnji četvrti skup ScanNet je sadržavao snimke zatvorenih prostorija.

Razvili smo prototipnu proceduru za učenje modela koja dobro generalizira na sva četiri skupa za testiranje. Skupovi se jako razlikuju osim domene i u broju primjera te veličini slika. WildDash skup sadrži teške primjere s distorzijama i izvan distribucijske primjere što ovaj skup podataka čini najtežim od tri skupa vožnje. ScanNet sadrži mnoštvo slika zatvornog prostora koje su jako teške za segmentaciju budući da objekti imaju visoku varijancu u veličini, rotaciji, vidljivosti te količini prekrivanja. Pored toga oznake su vrlo neprecizne i sadrže dosta anotacijskih grešaka. Zbog navedenih razloga točnost segmentacije je najlošija upravo na ScanNet skupu. Jedan od bitnih faktora prilikom učenja je miješanje primjera iz svih skupova za učenje u svakoj mini-grupi. Rezultati pokazuju jako malo preklapanja u pogrešno klasificiranim pikselima između tri skupa vožnje i ScanNeta. To upućuje da je model implicitno naučio razlikovati dvije domene.

Prikazana metoda zasniva se na efikasnoj arhitekturi predstavljenoj u prethodnom poglavlju utemeljenoj na gusto povezanim slojevima za raspoznavanje te ljestvičastog puta za naduzorkovanje.

Šesto poglavlje: Zaključci

U ovom poglavlju prikazan je sažetak rezultata i doprinosa ostvarenih tijekom rada na ovoj doktorskoj disertaciji. Osnovne prednosti predloženih metoda su prostorna i vremenska efikasnost i visoka točnost za problem semantičke segmentacije.

Ključne riječi: računalni vid, duboko učenje, semantička segmentacija, konvolucijske neuronske mreže, Cityscapes, Pascal VOC2012, CamVid, KITTI

Contents

Chapter 1

Introduction

1.1 Learning to see

The vision is perhaps the most important sensory experience for us humans. We navigate our world primarily with vision. We use it to solve everyday tasks. Even our dreams are visual. It is no wonder then if we want to offload some of our vision-based work to computers, we first need to enable them to see. The goal of the field of computer vision is to endow the computers with vision. Applications of the field to the real world problems are numerous. Some of them are autonomous navigation of robots and vehicles, medical imaging, safety and surveillance systems, biometric identification, content-based image retrieval, consumer electronics, video editing and movie production, scene reconstruction for 3D maps, cultural heritage, etc. But as much as vision comes easy to us humans, it is nevertheless one of the hardest problems to solve in computer science. Digital image is just a collection of pixel values representing the amount of photons that hit each camera sensor cell during exposure. Yet somehow, we must be able to extract high-level semantic information from those pixel values. We know that because we do it ourselves all the time. The image contains high-dimensional unstructured data and our objective is to model complex functions that will map that data to useful predictions. These predictions should be structured and rich in semantics. For instance, if we want to recognize a dog in the image the output of our function can be just a 0 or 1, depending upon the presence of dog in the image. For many vision problems the best solutions are obtained by learning instead of handcrafting. This is achieved through machine learning algorithms whose goal is to enable computers to learn how to solve a task without being explicitly programmed to do so. Recent advances in the field brought an impressive progress, in large part thanks to the renaissance of deep convolutional models. In this thesis we shall see how convolutional neural networks work when applied to the problem of *semantic segmentation*.

1.2 Semantic Segmentation

Semantic segmentation is an exciting computer vision task with many potential applications in robotics, intelligent transportation systems, medical imaging, photo editing, etc. In semantic segmentation a trained model classifies pixels into meaningful high-level classes. The goal is to associate each image pixel with a high-level class such as the sky, a tree or a person. Semantic segmentation provides rich information on surrounding environment, which presents clear application potential in many domains. However, there are challenges which are still to be solved before this exciting technique can be fully deployed for the real world problems.

Most of the recent successful approaches in the field rely on the convolutional neural networks trained as dense multi-class classification models, usually in the supervised learning framework. Thus, our models are trained on pairs of input examples and annotated target labels. Figure 1.1 shows an example of the model input and the desired output. The input is an image obtained from the camera while the target is a dense semantic map in which colors correspond to classes. The model has to produce the prediction of dense pixel labels. It is common to encode this prediction as a 3-dimensional tensor of categorical distributions, each distribution corresponding to one pixel location. The first two dimensions H and W correspond to the height and width of an image while the third dimension C represents class probabilities. Our training objective is a scalar function of dense predictions and target labels, which is defined as a mean of non-negative per-pixel contributions. Each per-pixel contribution is proportional to the mismatch between the particular prediction and the corresponding target. Hence, we often denote our training objective as loss. The training procedure optimizes the model parameters in order to minimize the loss.

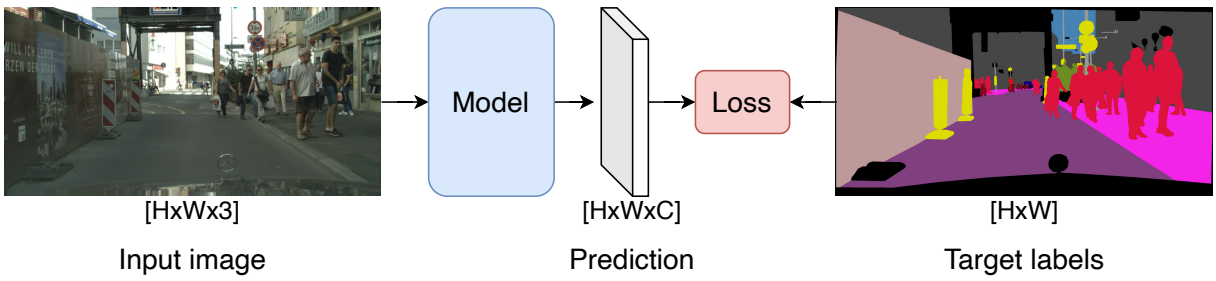

Figure 1.1: The goal of the semantic segmentation model is to associate each image pixel with the corresponding high-level label. In the fully supervised setup the model is trained on the dataset of pairs of input images and annotated target labels. During training we minimize the mismatch between model predictions and target labels. The predictions are encoded as a tensor of pixel-level posterior class probabilities $P(y_{ij} = c|x)$. H and W denote the height and width of the image while C denotes the number of target classes.

Due to being complementary to the more famous object detection problem, semantic segmentation represents an important step towards advanced image understanding. Figure 1.2 presents the difference between semantic segmentation and object detection. Object detection has to regress bounding boxes around each instance of the known set of classes. These bounding boxes are designated with white rectangles in Figure 1.2. Each bounding box denotes a unique instance of a known class and has a well defined spatial range aligned with the object boundaries. On the other hand, in semantic segmentation the class of the pixel is determined by the object or stuff projected to that concrete pixel. Hence, the object size is not encoded in the model prediction as in object detection where size is defined by the bounding box. Nevertheless, the model will need to look at the appropriate neighbourhood of a particular pixel in order to classify it.

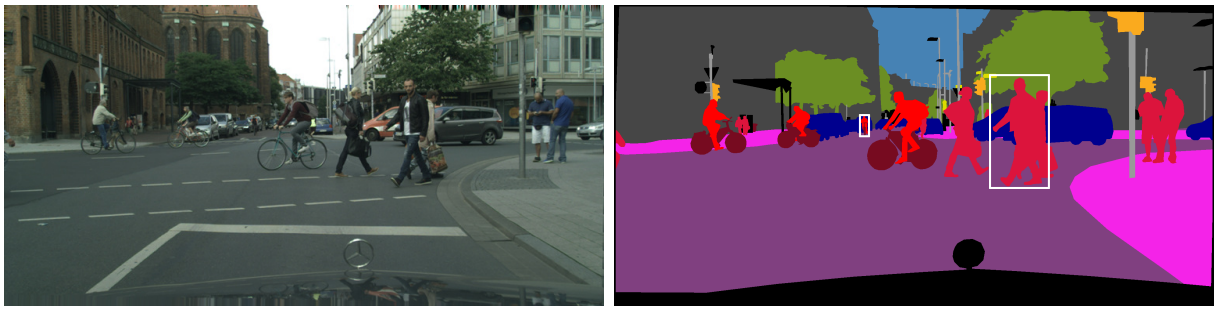

Figure 1.2: Difference between semantic segmentation and object detection. In object detection we only need to predict the bounding boxes around objects of interest. In semantic segmentation usually almost all pixels need to be classified to corresponding classes. Note that context size needed to classify each pixel exhibits a large variance in size due to a wide range of scale at which objects appear.

1.3 Challenges

There are numerous challenges that emerge while applying convolutional neural networks to semantic segmentation of large images. Classical convolutional networks were designed for the problem of image classification. In image classification the goal is to classify only the occurrence of objects in images. Thus, all information about object location is lost in the prediction. On the other hand, semantic segmentation targets dense labels. Consequently, the problem arises of how to adapt the convolutional neural network for this kind of prediction. Let us now define the minimal context as the area of neighbouring pixels needed to classify a particular pixel. Due to objects and stuff exhibiting a large variance in the scale they appear in the image, the problem arises of determining the appropriate context size. For each pixel the context size will be different depending on the scale of the object it belongs to. This is depicted in the Figure 1.2 with two bounding boxes. The person nearby will need a much larger context size compared to the distant

person in order to classify all of its pixels correctly.

Figure 1.3 shows another more extreme case of a large context size needed to classify the green pixel belonging to the white truck. Let us now define the concept of the model receptive field as the maximum context size that affects a particular model decision. A typical convolutional network designed for image classification will only be able to perceive the context inside the blue square which is the limit of its receptive field. However, the context needed to classify the green pixel is much bigger as shown by the red rectangle. Therefore, we will need to find a way to increase the receptive field of the model by a large factor. This is an important challenge that we will need to overcome while making convolutional models suitable for semantic segmentation of large images. Furthermore, due to the large input images we also need to address model efficiency, both in inference time and memory requirements. In the following chapters, we present solutions to these challenges.

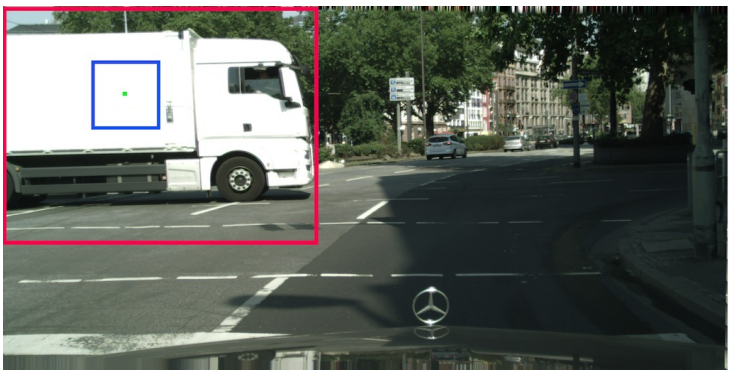

Figure 1.3: Convolutional networks designed for image classification have limited receptive field shown inside the blue square. The ideal context size needed to classify the green pixel is shown inside the red rectangle.

1.4 Contributions

The main contributions of this thesis are as follows. First, we develop a scale-invariant convolutional model for semantic segmentation which alleviates the problem of learning the same object on a wide range of scales. We increase the accuracy of the model by developing a scale selection technique based on image pyramid and stereoscopic depth information. Second, we contribute a new RGB-D dataset for semantic segmentation of driving scenes. The dataset contains groundtruth semantic segmentations and stereoscopic depth images for 445 annotated hand-picked camera images from the KITTI dataset. We start from 146 images annotated by German Ros et al. [2], improve their annotation accuracy and contribute another 299 images. Third, we develop an efficient architecture for dense prediction on large images based on a densely connected feature extractor and lightweight

ladder-style upsampling. The presented architecture is computationally and memory efficient and achieves a great tradeoff between inference speed and generalization accuracy. Fourth, we present the results from the Robust Vision Challenge 2018 where we achieved the second place. The goal of the competition was to assess the robustness of the model to multiple dataset domains. We have found out that mixed batches ensure the most stable evolution of batchnorm parameters. Low incidence of foreign classes indicates that our models succeeded to implicitly learn to distinguish the domains.

To summarize, the contributions of this thesis are as follows:

1. efficient asymmetric architecture for semantic segmentation consisting of a custom densely-connected recognition module, lightweight spatial pyramid pooling and lightweight upsampling (Chapter 4);
2. module for extracting scale-invariant convolutional representations based on dense scale selection with respect to local stereoscopic depth (Chapter 3);
3. methodology for learning semantic segmentation models on multi-domain datasets (Chapter 5).

1.5 Thesis structure

The remainder of this thesis is structured as follows. Chapter 2 presents an overview of deep convolutional models and how to adapt models designed for the image classification problem to be well suited for the problem of semantic segmentation. Chapter 3 presents a novel convolutional architecture which allows to recover scale-invariant convolutional representations. That work was accepted for oral presentation at GCPR 2016 [1]. Chapter 4 presents a novel convolutional architecture for dense prediction which allows correct classification of pixels at both large and small objects by blending features taken from the different layer depth. These features have different tradeoffs between location accuracy and context size, allowing the model to correctly classify objects appearing on a wide range of scales. In contrast to previous symmetric encoder-decoder approaches, our design features a thick encoder and thin decoder. That work was accepted for presentation at the CVRSUAD workshop at ICCV 2017 [4]. Chapter 5 presents an application of findings from Chapter 4 in our submission to the Robust vision challenge in 2018 which took the 2nd place among 10 submissions. Contributions from Chapter 4 and Chapter 5 have been published in IEEE Transactions on Intelligent Transportation Systems [3].

Chapter 2

Deep Convolutional Models

This chapter provides the overview of deep convolutional models and main challenges in applying them to the problem of semantic segmentation. First, it gives the necessary theoretical foundations from machine learning used in the thesis. Second, it introduces convolutional neural networks and explains how they were first applied for the image classification problem. Finally, we discuss the challenges of adapting convolutional networks to semantic segmentation and give an overview of approaches.

2.1 Supervised Learning

In the supervised learning scenario the model f is trained on the dataset of examples x associated with target values y. The goal is to learn a function f that performs mapping $f : X \mapsto Y$. Supervised learning assumes that we can collect a large enough dataset of (x, y) pairs. If the process of data collection and annotation is not too expensive, than supervised learning is a feasible approach to solving the problem. We commonly use supervised learning for the types of problems where it is hard or seemingly impossible to manually program the function f. This is the case for semantic segmentation and many other practical problems in computer vision.

Semantic segmentation belongs to the broad category of classification problems in the field of machine learning. In a classification problem the target values y are commonly referred to as labels. Each label represents one of the classes that the model is expected to recognize. The input to the semantic segmentation model f is an RGB image of shape $3 \times h \times w$ usually normalized to have zero-mean and unit-variance. More formally, $x \in \mathbb{R}^d$, $y \in \{1, ..., k\}$ and $f : \mathbb{R}^d \mapsto \{1, ..., k\}$ where input dimensionality $d = 3hw$ and k denotes the number of classes.

In order to learn the model function f defined by parameters θ we would like to minimize the expected loss $J^*(\theta)$ over data generating distribution p_{data} or more formally:

$$J^*(\theta) = \mathbb{E}_{(x,y)\sim p_{\text{data}}} L(f(x;\theta), y) \tag{2.1}$$

where $f(x;\theta)$ is the predicted output from the model on the input example x and θ denotes all trainable parameters of the model.

However, in practice it is not tractable to compute the expectation over data generating distribution because there are to many possible (x, y) examples. Instead, we have access only to data we managed to collect. Therefore, we approximate the loss function by taking the average loss across all training data:

$$J(\theta) = \frac{1}{N}\sum_{i=1}^{N} L(f(x_i;\theta); y_i) \tag{2.2}$$

The assumption here is that the training data is independent and identically distributed (i.i.d.). Finally we can represent the learning as an optimization problem where the goal is to find model parameters θ^* that minimize the loss function given in Equation 2.2:

$$\theta^* = \arg\min_{\theta} \frac{1}{N}\sum_{i=1}^{N} L(f(x_i;\theta), y_i) \tag{2.3}$$

While the optimization goal is to minimize the loss function on the training data the ultimate goal of learning is to find the model which generalizes well on unseen data. Given that the space of parameters θ is usually very large, there will exist many model instances that achieve low loss value on training data. However, their generalization ability on unseen data can vary greatly. In order to steer the optimization method towards solutions θ^* that generalize better we can use a technique called regularization. The goal of regularization is to adjust either the loss function $J(\theta)$, model function f or optimization method in such a way that it promotes solutions θ^* that generalize better among all possible solutions. The most popular way of achieving regularization is by adding additional regularizing term to the loss function. For instance, if f is some type of a neural network then we usually use L2 regularization. In L2 regularization we add the squared L2 norm of the parameters θ to the loss term. The new regularized loss takes the following form:

$$J(\theta) = \left[\frac{1}{N}\sum_{i=1}^{N} L(f(x_i;\theta), y_i)\right] + \lambda \left\|\theta\right\|_2^2 \tag{2.4}$$

where λ is the hyperparameter describing the regularization strength.

2.2 Optimization

Optimization refers to the process of minimizing or maximizing the objective function. In our case the objective function is the loss function from Equation 2.4 $J(\theta)$ that we want to minimize. In cases when f is a deep neural network, the loss function can not be minimized with a closed form solution. Therefore, deep models like convolutional networks are optimized with first order methods. Second order methods are not feasible because the Hessian matrix of second order derivatives is computationally too expensive to calculate due to the large number of parameters. First order methods utilize first order partial derivatives of the loss function $J(\theta)$ with respect to model parameters θ. The obtained vector of derivatives is also known as the gradient, denoted as $\nabla_\theta J(\theta)$. Note that in order to apply gradient based optimization, the functions L and f have to be differentiable. The simplest and most famous first order optimization method used in machine learning is the stochastic gradient descend or SGD, outlined in Algorithm 1.

Algorithm 1 Stochastic gradient descent (SGD).

Input Learning rate ϵ, batch size m, regularization strength λ
Input Initial model parameters θ
repeat
1. Sample a minibatch of m examples from the training set $\{(x_1, y_1), ..., (x_m, y_m)\}$
2. Compute the gradient: $g \leftarrow \nabla_\theta J(\theta) = \nabla_\theta \left[\frac{1}{m} \sum_{i=1}^{m} L(f(x_i; \theta), y_i) + \lambda \|\theta\|_2^2\right]$
3. Apply parameter update: $\theta \leftarrow \theta - \epsilon g$

until stopping criterion is satisfied

The main idea behind the gradient descent is to move in the negative direction of the gradient. We want to go in the negative direction because the gradient shows the direction where the loss function is increasing and we want to minimize the loss. However, we must be very careful not to overstep during parameter update. If the update step is too large, we can easily move to the parameter space with higher loss value. The reason behind this is that the gradient can only give us a linear approximation of the loss function which is in fact highly non-convex. Therefore, the most important hyperparameter of gradient descent methods like SGD is the learning rate ϵ. If the learning rate is too high, the optimization will not converge and may even diverge. If it is too low the optimization will be too slow and we may not have the time or resources to wait it out. Furthermore, it would be too expensive to compute the gradient g on the entire training set in each step. In SGD the idea is to sample a random subset of examples in each step and compute the gradient approximation on that small sample. This trick allows the SGD to iterate and converge much faster. The random subset of examples is called the minibatch or batch and the size of the sample is defined by the hyperparameter m also known as batch size.

2.3 Deep Convolutional Models

Deep convolutional neural networks have caused an unprecedented rate of computer vision development. They were first introduced by LeCun et al. [5] in 1998 but got the attention of the whole community in 2012 when Krizhevsky et al. [6] participated in the greatest image classification challenge at the time (ImageNet [7]). Their submission won by a very large accuracy margin compared to the second place solution.

The main idea behind convolutional networks is the focus on unique kind of spatial topology where we want our model to be invariant or equivariant to the translation. This turns out to be especially well suited for images where the same object can easily occur at any position in the image. In comparison, fully connected networks don't work well on image data because they make less assumptions about data. If we connect each neuron with all the pixels in the image then they would be forced to learn specific features only on particular position in the image. Therefore, the inductive bias of convolutional models is a much better choice for images.

A typical convolutional network has four main types of layers:

- Convolution
- Non-linearity
- Subsampling
- Normalization

The input to the network is a 4 dimensional tensor of shape $N \times C_1 \times H_1 \times W_1$. N is the number of image examples in a batch, also known as the batch size. H_1 and W_1 and height and width of the image and C_1 is the number of color channels in the image (usually 1 or 3). For simplicity, in the examples later in this chapter we assume $N = 1$ case where only a single input image processed. However, training the models with large enough N is important in practice as it makes the gradients more stable and because normalization layers usually depend on N. The input image tensor of shape $C_1 \times H_1 \times W_1$ is first processed by the first convolution layer. This is illustrated in the Figure 2.1. The convolution layer consists of a set of convolutional filters. Let us denote the number of filters in the first convolution layer with C_2. Each filter has a shape $K_1 \times K_1 \times C_1$ and will perform a convolution across the height and width of the input. The output of the convolution is another tensor with the shape of $C_2 \times H_2 \times W_2$ where $H_2 = H_1 - K + 1$ and $W_2 = W_1 - K + 1$. Note that $K - 1$ pixels of resolution are lost on both dimensions due to the border limits. The values in the output tensor are often referred to as the layer activations. Typical values for the filter size K are between 1 and 7.

Due to some models having many convolutional layers, sometimes even hundreds, a large decrease of resolution would accumulate. More specifically, $L(K-1)$ pixels would be lost on each dimension where L is the number of convolutions. Therefore, in most

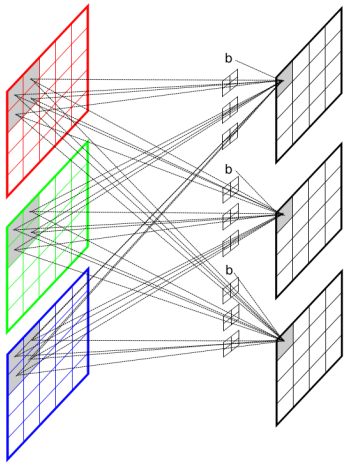

Figure 2.1: Visualization of of the convolution operation. The input RGB image is convoloved with three filters of size 2×2. At each position the dot product between the filter and the input values is first computed after which the bias term b is added. Image reproduced from [8].

cases we would like to preserve the spatial shape of the input tensor on the output and have $H_2 = H_1$ and $W_2 = W_1$. This is achieved by padding the input tensor at each of the 4 spatial borders. The number of padded pixels we need to add to each border is $P = (K - 1)/2$. Most commonly used padding is zero padding where all values in the added dimensions are set to zero.

After convolution, we apply the non-linearity operation to all values of the tensor. Typical non-linearity operations are rectifed linear unit (ReLU), sigmoid function and the tanh function. Adding non-linearity between convolutions is a very important feature of convolutional models. Without non-linearity, multiple subsequent convolutions would always define a linear function, no matter how many convolutions we stack together. This occurs since the composition of multiple linear functions is also a linear function. Non-linearity layers break the linear relation between convolution layers and enable the model to capture a complex high-dimensionality manifolds capable of describing a complex input data like images of natural scenes. The output tensor from non-linearity layer is used as the input tensor to the next convolution layer. This process is then repeated for all convolution and non-linearity layers. More generally, the convolutional networks have a directed acyclic graph (DAG) topology where nodes represent the operations and edges represent the flow of tensors during graph evaluation. Each operation or node can take multiple inputs but produce only one output. In the basic convolutional network all nodes have exactly one input and one output.

The LeNet architecture introduced by LeCun et al. [9] in 1998 is the convolutional neural network for character recognition that had a large influence on deep learning revolution which started after AlexNet [6] won the ImageNet challenge in 2012. The LeNet model is illustrated in Figure 2.2. The model was designed for the MNIST dataset for

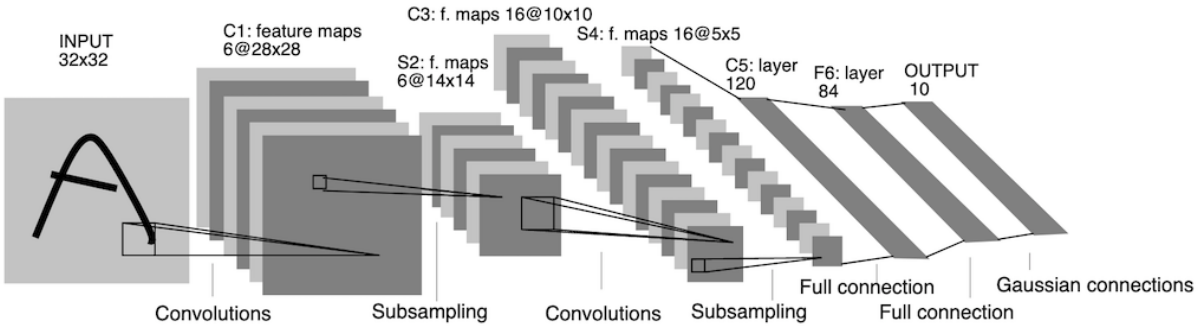

Figure 2.2: The LeNet architecture introduced by LeCun et al. [9]. It features 3 convolutional layers, 2 max-pooling layers and a fully connected layer at the end. All three convolutions have filter size of 5×5. Note that last convolutional layer behaves like a fully connected one because the size of the input feature maps is the same as the filter size (5×5). Figure reproduced from [9].

handwritten digit recognition. The MNIST dataset has images of size 32×32 and 10 classes corresponding to digits. Due to that, the architecture of LeNet was rather simple compared to modern counterparts.

Let us now introduce the concept of the receptive field. The receptive field of a particular activation is composed of all the pixels in the input image that are used to compute that activation value. We can further define the receptive field of a particular layer as the size of the rectangle which defines the receptive field of any activation produced by that layer.

In the computational graph, each consecutive convolution will increase the receptive field size. However, if we only use vanilla convolution layers the size of the receptive field will grow very slowly. More formally, the size of the receptive field at the n-th convolutional layer is defined by the equation $k_n + \sum_{i=1}^{n-1}(K_i - 1)$. For instance, 50 convolutions each having $K = 3$ would produce the receptive field of size 101×101 pixels ($3 + 49 \cdot 2 = 101$). Unfortunately, this rate of growth is not feasible, even more so for the large input images we need to deal with.

In order to solve this problem we will now introduce the subsampling operation. In a subsampling operation the height and width of the input tensor are reduced by a factor of S. The effect of this spatial size reduction will be a dramatically increase the receptive field of all the following convolutions. Basically, the rate of growth in the layers that follow after subsampling will be multiplied by the factor of S. By combining multiple subsampling operations at different places in the graph we can easily obtain exponential increase in the receptive field size. The subsampling operation is usually achieved in one of two ways: by using a strided convolution or by applying the pooling operation. Strided convolution will have a step size of 2 or larger while iterating over input locations. For example, stride size of 2 will apply the convolution on sparse locations with the

distance of 2 pixels between adjacent locations. Pooling operation is very similar in a way that is also used a strided iteration but instead of applying the convolution it applies some fixed function of the input. Popular function choices are max and average pooling. We can draw a line between these two approaches by comparing strided convolution with average pooling operation. In average pooling the weights are fixed to always compute the average while the convolution has learnable weights and computes the weighted average of the inputs. While convolutions are equivariant to translation and increase the receptive field very incrementaly, subsampling layers introduce invariance to the translation in the input image and produce a large increase in receptive field. Subsampling operations produce the output tensor of shape $C_2 \times H_2 \times W_2$ where $H_2 = \lfloor (H_1 - K + 2P)/S \rfloor + 1$ and $W_2 = \lfloor (W_1 - K + 2P)/S \rfloor + 1$.

The goal of the normalization operations is to improve the stability of the optimization process, enable faster training. More often than not, this also leads to a better model generalization. By far the most used normalization operation is Batch Normalization (BatchNorm) [10]. In BatchNorm usually the output from the convolutional layer is normalized to have zero mean and unit variance. In the original paper the authors argue that BatchNorm reduces the internal covariate shift which is described as an effect of introducing change to the distribution of features caused by the updated to the preceding layers during training. The hypothesis is that this internal covariate shift has a negative impact on training but this still remains an open research problem. In the later work Santurkar et al. [11] argue that BatchNorm doesn't really reduce the internal covariate shift. They perform experiments to show that BatchNorm has a smoothing effect on the loss surface. This has stabilizing effect and reduces noise in gradients during training which allows for the well observed faster training and often better generalization. The main issue with BathNorm is its sensitivity to small batch size. Alternative methods like GroupNorm [12] are being developed to alleviate this issue. GroupNorm addresses this problem by grouping feature maps and than computing the mean and variance for a whole group instead of each feature map separately.

Deep convolutional models have caused an unprecedented rate of computer vision development. Their sudden rise to the focus of research attention has been catalized by rapid development of GPU computing, appearance of large image classification datasets [7], and simple but non-obvious additions [10, 13, 14, 15] to the early work in the field [9]. Model depth has been steadily increasing from 8 levels [6] to 19 [16], 22 [17], 152 [18], 201 [19], and beyond [18].

2.4 Image Classification

With all the building blocks of the convolutional network stacked in a simple feedforward manner it is straightforward to build an architecture suitable for image classification problem. In the image classification problem the input to the model is the image and the output is a vector which encodes a probability distribution over known classes. Typical ConvNet architecture for image classification on the ImageNet [7] problem is shown in Figure2.3. The number of feature maps in the intermediate tensors correspond to the DenseNet121 model [20].

First, an input image of size $224 \times 224 \times 3$ is processed by a stem unit where its resolution is subsampled 4 times usually by strided convolution and max-pooling operations. The obtained $56 \times 56 \times C$ tensor is then processed by 4 subsequent convolutional blocks. Each of the convolutional blocks consists of multiple units of stacked convolution, normalization and non-linearity layers. Furthermore, blocks 2, 3 and 4 have subsampling operations at the beginning where the input resolution is halved on both spatial axes. DenseNet uses average-pooling here. It is common to increase the number of feature maps after resolution subsampling. We can afford this because each subsampling reduces the number of pixels to process by 4 times so it is reasonable to transfer that computation to feature space. This will have a big impact on model capacity and typically last to blocks hold most of the parameters. Consequently, in our DenseNet121 case the 4 blocks output tensors with $[256, 512, 1024, 1024]$ feature maps. Therefore, the final fourth block produces feature tensor of shape $7 \times 7 \times 1024$. This feature tensor is then sent to the classification head (FC head) where first the global spatial pooling is applied to obtain only a vector of shape 1024 and then projected with a fully connected (FC) layer to the final logits. The logits vector represent per-class scores and in the ImageNet case is has 1000 elements, one for each class.

2.4.1 Loss Function

In order to train image classification model we need to first define a loss function. For classification we use the probabilistic framework which allows us to model the uncertainty of the observed data. Furthermore, we can utilize the maximum likelihood estimation (MLE) from probability theory to get the point estimate of the model parameters. Under MLE we define a loss function with respect to the model parameters which maximizes the likelihood of observing the data in the training set.

First step in defining a loss function is to map the predicted logits from our model to probabilities. We need a differentiable function which will normalize the logits in a way that they sum to 1 and that they are non-negative. This will transform logits to represent

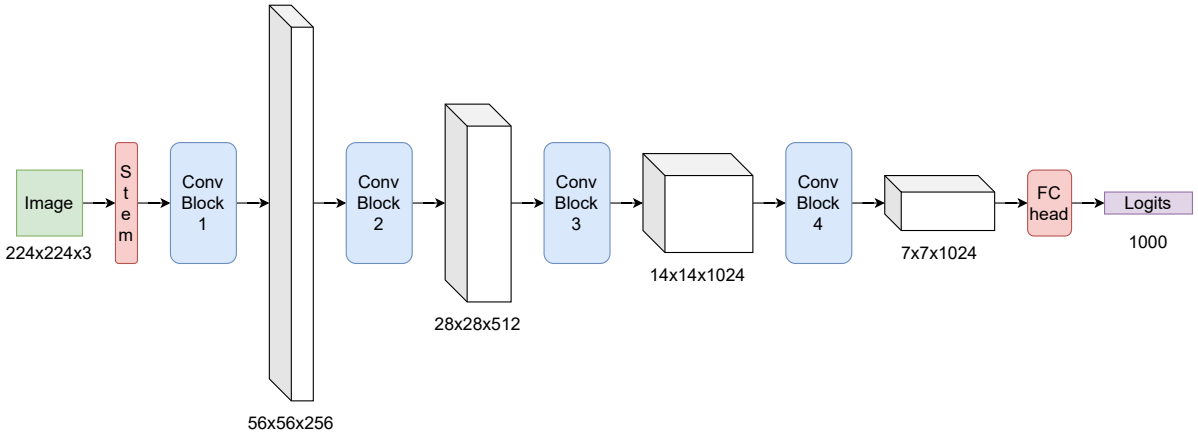

Figure 2.3: Image classification model architecture for ImageNet. All building blocks are shown, but the focus is on the convolutional blocks and their output tensors. Tensor shapes are in $H \times W \times C$ format and correspond to the DenseNet121 model [20].

valid probability distribution over classes. The most common function used to achieve this is the softmax function which takes a input vector of logits $\mathbf{z}$ and outputs a vector of probabilities $\mathbf{p}$. It is defined in the equation below

$$p_i = \sigma(\mathbf{z})_i = \frac{e^{z_i}}{\sum_{k=1}^{K} e^{z_k}} \tag{2.5}$$

where K is the number of classes and p_i the probability of the i-th class or $\mathbf{p}[i]$.

The most commonly used loss function for classification is the cross-entropy loss, which is defined as:

$$L(\mathbf{p}, \mathbf{y}) = -\sum_{k=1}^{K} y_k \log p_k \tag{2.6}$$

Cross-entropy comes from information theory and in our case can be viewed as an non-negative distance between two distributions $\mathbf{p}$ and $\mathbf{y}$. Most often, the true distribution $\mathbf{y}$ is set to be a one-hot vector putting all probability mass on the correct class. In that case the loss can be further simplified to negative log probability of the correct class:

$$L(\mathbf{p}, \mathbf{y}) = -\log p_k \quad [y_k = 1] \tag{2.7}$$

where k is the index of the correct class. Under the assumption that the examples in the dataset are independent and identically distributed (i.i.d.) by minimizing this loss we arrive to the maximum likelihood estimation of the model parameters θ^*:

$$\theta^* = \arg\min_{\theta} \frac{1}{N} \sum_{i=1}^{N} L(f(\mathbf{x}_i; \theta), \mathbf{y}_i) = \arg\min_{\theta} \frac{1}{N} \sum_{i=1}^{N} -\log p_k \tag{2.8}$$

where $f(\mathbf{x}_i;\theta)$ is prediction $\mathbf{p}_i$ from the model f given input example $\mathbf{x}_i$.

2.5 ResNet architecture

ResNet architecture [18, 21] introduced skip-connections to the classic neural network architecture composed of stacked feed-forward layers (Figure 2.5). The authors observed that for classic neural networks, the gradient descent optimization method is very sensitive to the model depth. If we add too much layers the optimization becomes unstable resulting in a worse error on both train and test datasets. This suggests that the problems of training very deep models are not caused by overfitting due to higher model complexity. Instead, they conclude that the optimization algorithm is unable to locate a good minimum on the loss surface. This hypothesis is further investigated by Li et al. in [22] where they arrive to the same conclusion by visualizing the loss surface of different convolutional architectures.

In order to fix these problems, Kaiming He has proposed to sum the output of each convolutional unit with its input via a skip-connection, as illustrated in Figure 2.5. As a result, the subsequent convolutional unit has contact with units which are not its immediate predecessors. This kind of skip-connection is called a residual connection, since each convolutional unit is encouraged to learn an additive correction factor (or residuum) of the preceding part of the convolutional model. Adding the input features of the convolutional unit to the output features helps the optimization procedure to find a better solution in the parameter space as we increase the number of layers. Basically, with residual connection the convolution has to only learn the residual or the difference between the input features and desired output features. This will encourage the optimization procedure to learn the identity function which will enable to model to propagate good features in early layers. Classic architectures have a very hard time learning the identity function. For example, it would have to learn the convolution filter where the center weight is 1 and all the rest are 0. With residual connection it only has to learn the filter where all the values are set to 0. Considering that random initialization sets most weights close to zero and with L2 regularization pushing all weights to zero, residual connections make an identity function very easy to learn. Because of these properties, residual connections enable the deeper ConvNet models to achieve lower train and test errors and generalize better than shallow models.

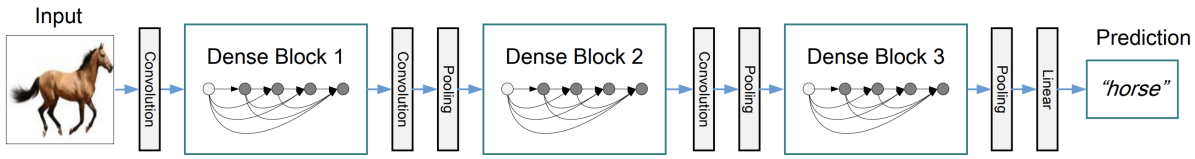

Figure 2.4: DenseNet with three dense blocks for an image classification task. Figure reproduced from [20].

2.6 DenseNet architecture

DenseNet architecture [20] addresses the same problem we discussed with ResNet, that is, making the deep convolutional models trainable with stochastic gradient descent. In DenseNet the output of a convolutional layer is concatenated with the outputs of all preceding layers, as illustrated in Figure 2.6. The obtained tensor is then used as an input to the next layer. In this way we obtain dense connectivity between layers while preserving the feed-forward processing. In contrast to ResNet, DenseNet combines features through concatenation instead of summation. The main advantage of DenseNet is that it does not need to learn to propagate already strong features in earlier layers through all subsequent layers. Because of its heavy reuse of previous feature maps, DenseNet requires fewer parameters than ResNet for the same model depth which we discuss in 2.7.4. DenseNet can be viewed as a subset of ResNet where a large part of the weights in each layer is fixed to identity function. Therefore, we can say that DenseNets behave like regularized ResNets in a way such that the feature propagation is enforced, while requiring much less computational resources due to not having to perform the actual propagation.

2.7 Comparison between ResNet and DenseNet

Most convolutional architectures [16, 17, 20, 21] are organized as a succession of processing blocks which process image representation on the same subsampling level. Processing blocks are organized in terms of convolutional units [21] which group several convolutions operating as a whole.

We illustrate similarities and differences between ResNet and DenseNet architectures by comparing the respective processing blocks, as shown in Figures 2.5 and 2.6. We consider the most widely used variants: DenseNet-BC (bottleneck - compression) [20] and pre-activation ResNets with bottleneck [21].

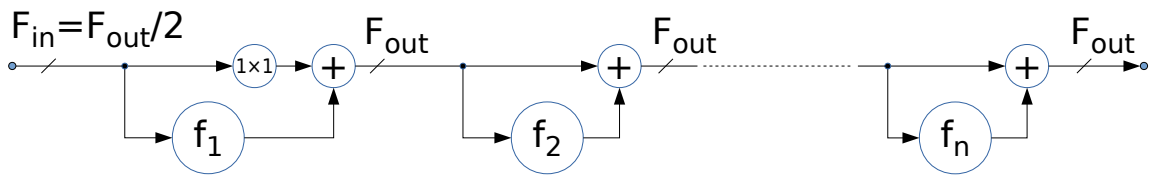

Figure 2.5: A pre-activation residual block [21] with n units. Labeled circles correspond to convolutional units (f_1-f_k), summations (+) and n-way concatenations - concat (n). All connections are 3D tensors D×H×W, where D is designated above the connection line (for simplicity, we assume the batch size is 1). F_{in} and F_{out} denote numbers of feature maps on the processing block input and output, respectively.

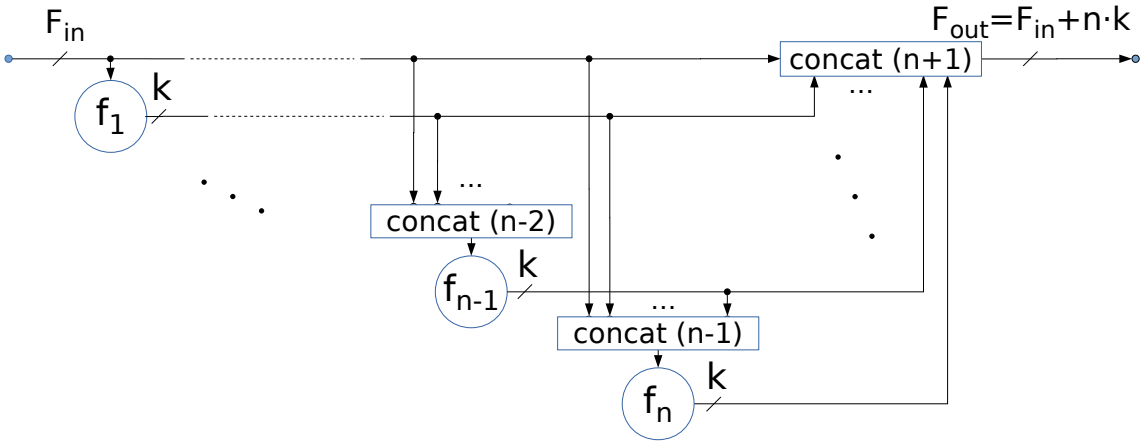

Figure 2.6: A densely connected block [20]. Labeled circles correspond to convolutional units (f_1-f_k), summations (+) and n-way concatenations - concat (n). All connections are 3D tensors D×H×W, where D is designated above the connection line (for simplicity, we assume the batch size is 1). F_{in} and F_{out} denote numbers of feature maps on the processing block input and output, respectively.

2.7.1 Organization of the processing blocks

The main trait of a residual model is that the output of each convolutional unit is summed with its input. This is illustrated in Figure 2.5. Hence, all units of a residual block have a constant output dimensionality F_{out}. Typically, the first ResNet unit increases the number of feature maps and decreases the resolution by strided projection. On the other hand, DenseNet units operate on a concatenation of the main input with all preceding units in the current block This is illustrated in Figure 2.6. Thus, the dimensionality of a DenseNet block increases after each convolutional unit. The number of feature maps produced by each DenseNet unit is called the growth rate and is defined by the hyper-parameter k. Most popular DenseNet variations have k=32, however we also consider DenseNet-161 with k=48.

In order to reduce the computational complexity, both ResNet and DenseNet units reduce the number of feature maps before 3×3 convolutions. ResNets reduce the dimen-

sionality to $F_{\text{out}}/4$, while DenseNets go to $4k$. DenseNet units have two convolutions (1×1, 3×3), while ResNet units require three convolutions (1×1, 3×3, 1×1) in order to restore the dimensionality of the residual datapath. The shapes of ResNet convolutions are: $1{\times}1{\times}F_{\text{out}} \times F_{\text{out}}/4$, $3{\times}3{\times}F_{\text{out}}/4 \times F_{\text{out}}/4$, and $1{\times}1{\times}F_{\text{out}}/4 \times F_{\text{out}}$. The convolutions in i-th DenseNet unit are $1{\times}1{\times}[F_{\text{in}}{+}(\text{i-1}){\cdot}\text{k}]{\times}4\text{k}$, and $3{\times}3{\times}4\text{k}{\times}\text{k}$. All DenseNet and pre-activation ResNet units [21] apply batchnorm and ReLU activation before convolution (BN-ReLU-conv). On the other hand, the original ResNet design [18] applies convolution at the onset (conv-BN-ReLU), which precludes negative values along the skip connections.

2.7.2 Time complexity

Both architectures encourage exposure of early layers to the loss signal. However, the distribution of the representation dimensionality differs: ResNet keeps it constant throughout the block, while DenseNet increases it towards the end. We compare a ResNet block (Figure 2.5) with its DenseNet counterpart (Figure 2.6) with the same hyperparameters F_{in} and F_{out}. DenseNet units have the following advantages: i) producing fewer feature maps (k vs F_{out}), ii) lower average input dimensionality, iii) fewer convolutions per unit: 2 vs 3. Asymptotic per-pixel time complexities of DenseNet and ResNet blocks are respectively: $O_{\text{DN}}(k^2n^2){=}O(F_{\text{out}}^2)$ and $O_{\text{RN}}(nF_{\text{out}}^2)$. Popular ResNets alleviate asymptotic disadvantage by having only $n{=}3$ units in the first processing block which is computationally the most expensive due to largest resolution. Hence, the temporal complexity of DenseNet models is only moderately lower than comparable ResNet models, although the gap increases with capacity [20].

We illustrate this by comparing ResNet-50 ($26{\cdot}10^6$ parameters) and DenseNet-121 ($8{\cdot}10^6$ parameters). On ImageNet test set the ResNet-50 achieves 76.1% Top1 accuracy while DenseNet-121 achieves 75.0%. Theoretically, ResNet-50 requires around 33% more floating operations than DenseNet-121 [20]. In practice the two models achieve roughly the same training speed on GPU hardware due to popular software frameworks being unable to implement concatenations without costly copying across GPU memory. Furthermore, popular GPU hardware is still unable to sustain the required bandwidth of 1x1 convolutions in DenseNet units [23]. However, in inference the DenseNet-121 is slightly faster than ResNet-50 after optimizing the costly concatenations. This optimization is already implemented in popular inference libraries like TensorRT. Note however that we have not considered improvements based on learned grouped convolutions [24] and early-exit classifiers [25].

2.7.3 Extent of caching required by backprop

Although DenseNet blocks do not achieve decisive speed improvement for small models, they do promise substantial gains with respect to training flexibility. Recall that backprop requires caching of all convolution input tensors in order to be able to calculate gradients with respect to convolution weights. This caching may easily overwhelm the GPU memory, especially in the case of dense prediction in large images. The ResNet design implies a large number of feature maps at input of each convolutional unit (cf. Figure 2.5). The first convolutional unit incurs a double cost despite receiving half feature maps due to 2^2 times larger spatial dimensions. This results in c_{RN} per-pixel activations which is clearly $O(n \cdot F_{\mathrm{out}})$:

$$c_{\mathrm{RN}} = (n+1) \cdot F_{\mathrm{out}} \,. \tag{2.9}$$

The DenseNet design alleviates this since many convolutional inputs are shared. An optimized implementation would need to cache only c_{DN} per-pixel activations:

$$c_{\mathrm{DN}} = F_{\mathrm{in}} + (n-1) \cdot k \approx F_{\mathrm{out}} \,. \tag{2.10}$$

Equations (2.9) and (2.10) suggest that a DenseNet block has a potential for n-fold reduction of the backprop memory footprint with respect to a ResNet block with the same F_{out}. We note that ResNet-50 and DenseNet-121 have equal F_{out} in the first three processing blocks, while ResNet-50 is twice as large in the fourth processing block.

Unfortunately, exploiting the described potential is not straightforward, since existing autograd implementations perform duplicate caching of activations which pass through multiple concatenations. Hence, a straightforward implementation has to cache the output of each unit in all subsequent units at least two times: i) as a part of the input to the first batchnorm ii) as a part of the input to the first convolution (1×1). Due to this, memory requirements of unoptimized DenseNet implementations grow as $O(kn^2)$ [26] instead of $O(kn) = O(F_{\mathrm{out}})$ as suggested by (2.10). Fortunately, this situation can be substantially improved with gradient checkpointing. The idea is to instruct autograd to cache only the outputs of convolutional units as suggested by (2.10), and to recompute the rest during the backprop. This can also be viewed as a custom backprop step for DenseNet convolutional units. We postpone the details for Section 4.5.

2.7.4 Number of parameters

It is easy to see that the number of parameters within a processing block corresponds to the number of per-pixel multiplications considered in 2.7.2: each parameter is multiplied exactly once for each pixel. Hence, the number of DenseNet parameters is $O(F_{\mathrm{out}}^2)$, while

the number of ResNet parameters is $O(nF_{\text{out}}^2)$. However, the correspondence between time complexity and the number of parameters does not remain the same at the model level since time complexity is linear in the number of pixels. Hence, per-pixel multiplications make a greater contribution to time-complexity in early blocks than in later blocks. On the other hand, the corresponding contribution to the number of parameters is constant across blocks. Therefore, the largest influence to the model capacity comes from the latter blocks due to more convolutional units and larger output dimensionalities.

Table 2.1 compares the counts of convolutional weights in ResNet-50 and DenseNet-121. We see that DenseNet-121 has twice as much parameters in block 1, while the relation is opposite in block 3. The last residual block has more capacity than the whole DenseNet-121. In the end, DenseNet-121 has three times less parameters than ResNet-50.

Table 2.1: Count of convolutional weights across blocks (in millions). ResNet-50 has n=[3,4,6,3] and F_{out}=[256,512,1024,2048] while DenseNet-121 has n=[6,12,24,16] and F_{out}=[256,512,1024,1024].

block@subsampling	B1@/4	B2@/8	B3@/16	B4@/32
Resnet-50	0.2	1.2	7.1	14.9
DenseNet-121	0.4	1.0	3.3	2.1

2.7.5 DenseNets as regularized ResNets

Let us consider a DenseNet block D and a ResNet block R with equal number of convolutional units n and an equal output dimensionality F_{out}. We will show that each DenseNet block D can be realized with a suitably engineered ResNet block R such that each R_i produces $F_{\text{out}} = F_{\text{in}} + nk$ maps. Assume that R_i is engineered so that the maps at indices $F_{\text{in}} + ik$ through $F_{\text{in}} + (i+1)k$ are determined by non-linear mapping r_i, while all remaining maps are set to zero. Then, the effect of residual connections becomes very similar to the concatenation within the DenseNet block. Each non-linear mapping r_i observes the same input as the corresponding DenseNet unit D_i: there are F_{in} maps corresponding to the block input, and $(i-1)k$ maps produced by previous units. Hence, r_i can be implemented by re-using weights from D_i, which makes R equivalent to D.

We see that the space of DenseNets can be viewed as a sub-space of ResNet models in which the feature reuse is heavily enforced. A ResNet block capable to learn a given DenseNet block requires an n-fold increase in time complexity since each R_i has to produce $O(kn)$ instead of $O(k)$ feature maps. The ResNet block would also require an n-fold increase of backprop caching and an n-fold increase in capacity following the arguments

in 2.7.3 and 2.7.4. We conclude that DenseNets can be viewed as strongly regularized ResNets. Hence, DenseNet models may generalize better when the training data is scarce.

2.8 Semantic Segmentation

In semantic segmentation the goal is to classify each pixel in the image to the kind of object or stuff it is a part of. Objects are all things with a well defined shape (e.g. cars, people) while stuff are all regions in the image without a specific shape (e.g. road, sky) [27]. This is different from image classification where to goal is to classify only the presence of usually only objects in the image. Consequently, in image classification we did not care about preserving the location information and could afford to loose it in subsampling layers. Now the objective is to modify the model in a way to preserve the location information and enable it to produce dense tensor of probabilities on the output. This tensor will encode the class probabilities for each pixel in the input image. Therefore, in the case of semantic segmentation the input to the model is an image and the output is a tensor of probabilities instead of only a single vector.

The most common evaluation metric used for assessing the quality of semantic segmentation model is the mean Intersection over Union (mIoU). It is defined as a mean over class IoU metrics. Basically, first the IoU is computed for each class separately and then the mIoU is obtained by taking the mean. Class IoU is defined as the intersection between the set of predicted segmentation pixels and the set of groundtruth pixels, divided by their union. Most datasets contain pixels which do not belong to any of the classes. They are usually labeled to a special *ignore* class and model prediction for those pixels is ignored during evaluation. More formally, mIoU is defined by the following equation:

$$\text{mIoU} = \frac{1}{C}\sum_{i=1}^{C}\frac{\text{TP}_i}{\text{TP}_i + \text{FP}_i + \text{FN}_i} \tag{2.11}$$

where C is the number of classes, TP_i denotes true positive, FP_i false positive and FN_i false negative predictions for the i-th class.

Early approaches to semantic segmentation optimized a trade-off between multiple local classification cues (texture, color etc), and their global agreement [28]. Later work improved these ideas with non-linear embeddings [29], multi-scale analysis [30], depth [1], and improved spatial consistency [31, 32]. However, none of these approaches has been able to match the improvement due to deep convolutional models [9, 30].

It is common for semantic segmentation to have the input image with a larger resolution compared to image classification, typically ranging from 0.5MP to 2MP. The question arises of how to convert an image classification model to one suitable for semantic seg-

mentation. Ideally, the converted model should keep all the good properties of the image classification model, mainly that it is computationally and memory efficient, has high model capacity and a large receptive field. There are two main approaches to achieving dense prediction in convolutional networks: dilated convolutions and ladder-style upsampling also known as feature pyramid network (FPN) [33]. Both approaches produce great results but offer different tradeoffs as we shall see in the following sections. More recently, a third approach called High-resolution network (HRNet) [34] has been proposed. It is based on designing an new architecture suitable for dense prediction instead of relying on the ImageNet based model. Before explaining the approaches for achieving dense prediction, let us first introduce the loss function for semantic segmentation.

2.8.1 Loss Function

Loss function for semantic segmentation is typically cross-entropy and the only difference to the image classification loss is that now we need to define a loss across all pixel positions. $\mathbf{p}$ and $\mathbf{y}$ are now 3-dimensional tensors instead of vectors. Therefore, after adding summation over height H and width W dimensions of the image the loss has the following form:

$$L(\mathbf{p},\mathbf{y}) = -\frac{1}{HW}\sum_{i=1}^{H}\sum_{j=1}^{W}\sum_{k=1}^{K} y_k^{(i,j)} \log p_k^{(i,j)} \tag{2.12}$$

2.8.2 Dilated Convolutional Models

The most popular way of adapting a classical ConvNet for a dense prediction task is to replace subsampling operations with dilated convolutions. Dilated convolutions [35] are convolutions which can achieve a large increase in the receptive field without utilizing subsampling operations. Figure 2.7 shows that dilated convolutions imply sparse instead of dense sampling of the input tensor. This is achieved by dilating the convolutional filter by an input stride factor r. Figure 2.7(a) shows a normal 3×3 filter ($r = 1$) applied on some feature map F_0. Red dots represent the filter sampling places and green area shows that the elements of output feature map F_1 have the receptive field of 3×3. In 2.7(b) the F_1 feature map is convolved by a dilated filter with $r = 2$. The result is a feature map F_2 with receptive field of 7×7. Finally, in 2.7(b)(c) dilated filter with $r = 4$ is applied to the F_2 and the output feature map F_3 now has the receptive field of 15×15. Thus, increasing r achieves exponential growth of the receptive field. Because the input is sampled at sparse locations with holes in-between them it is easy to tune the size of the receptive field by changing the size of the input stride of the dilated convolution. Note that without dilated convolutions the receptive field of the final F_3 feature map would be only 7×7.

Figure 2.8 shows the architecture of the semantic segmentation model obtained by

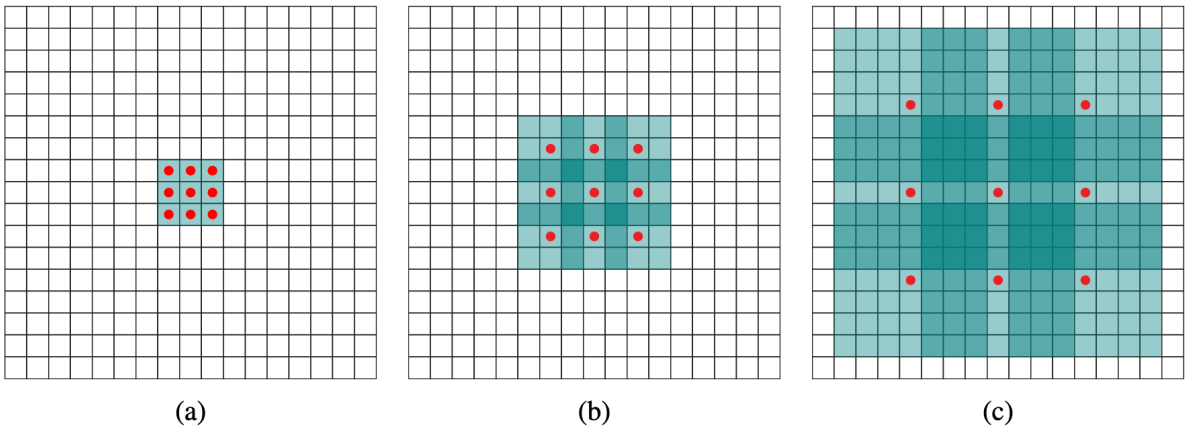

Figure 2.7: (a) normal 3×3 filter ($r = 1$), (b) dilated filter with $r = 2$, (c) dilated filter with $r = 4$. The figure is reproduced from [35].

introducing dilated convolutions in the last two convolutional blocks. The main objective is to obtain the dense prediction while keeping a large receptive field of the network.

The model from Figure 2.8 is very similar to the model from Figure 2.3. However all subsampling operations are removed and all subsequent convolutions are dilated with the previous input stride multiplied by hyperparameter r. Due to its similarity to the ImageNet model the dilated convolutional model is suitable for transfer learning because the segmentation model parameters can be initialized from the classification model trained on ImageNet. In order to add dilated convolutions, typically subsampling operation is removed and all convolutions after it are dilated with a value of r which preserves the receptive field. If some convolutions were already dilated before, the new dilation factor is obtained by multiplying the two factors. The dilated convolutions are employed by most of the recent works on semantic segmentation [36, 37, 38, 39, 40]. The upside of the dilated approach is that we do not need to introduce any new layers or parameters to the model. By dilating the convolutions and removing subsampling operations the density of the predictions is increased in a simple manner. The downsides are that dilated convolutions dramatically increase the computational and space complexity of the model. For the most classification models this means that they will become very memory inefficient during training and slow in inference time.

2.8.3 Dense Prediction with Ladder-style Upsampling

The problems of approach based on dilated convolutions can be solved by replacing them with ladder-style upsampling. Ladder-style upsampling, which we promote in this thesis [3, 4, 41], is based on the idea that we can recover the resolution lost due to subsampling by blending low resolution features in deeper layers with high resolution features from earlier layers. Deeper features are rich in semantics but lack the spatial resolution lost

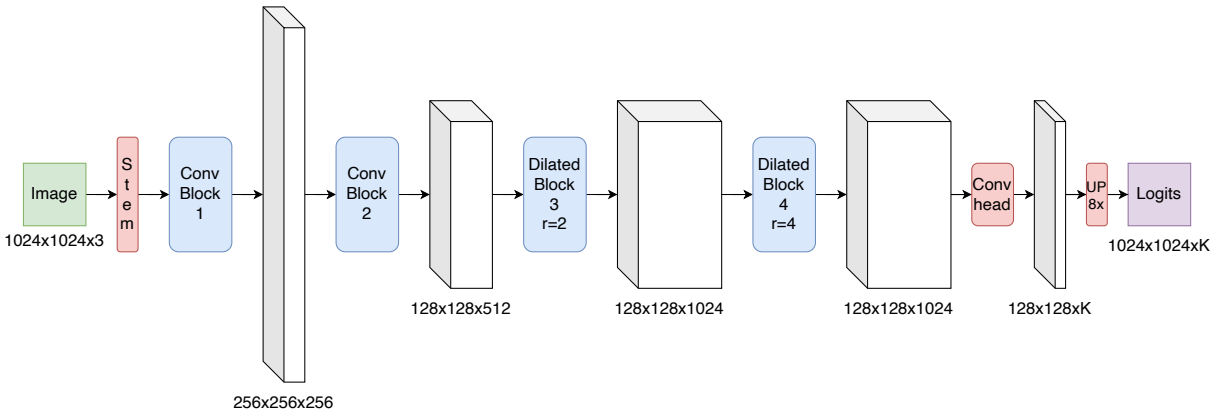

Figure 2.8: Semantic segmentation model obtained by utilizing dilated convolutions in the last two blocks. Note that the dilation rates that preserve the receptive field of the original model from Figure2.3 are $r = 2$ and $r = 4$. K denotes the number of classes on the output. Tensor shapes are in $H \times W \times C$ format and correspond to the DenseNet121 model [20].

during subsampling. Shallower features are rich in location accuracy and capable of capturing the semantics of smaller objects and low level details. This encourages the deep layers to discard location information and focus on abstract image properties. By combining features from all convolutional blocks we can achieve dense prediction and preserve all important properties of ConvNets. Furthermore, features from early layers encode small object details which are not easy to propagate through all the layers in the dilated approach and this propagation consumes model capacity. However ladder-style upsampling succeeds to recover most small objects lost during subsampling after blending with high-resolution features.

Figure 2.9 shows the architecture of a semantic segmentation model based on the ladder-style upsampling. The model input is an image of size 1024×1024 and the produced outputs are per-pixel logits encoded in the tensor of shape $1024 \times 1024 \times K$ where K is the number of classes in the dataset. The downsampling path (upper part) remains identical to the convolutional part of the classification model shown in Figure 2.3. The obtained feature tensor is then processed with the context unit. It is common the apply spatial pyramid pooling context unit [3] in order to capture long-range context information across the 32×32 grid. The resulting tensor is sent to the upsampling path (lower part). The upsampling path consists of the transition-up upsampling units (TU). Each upsampling unit takes the current tensor and blends it with the skip-connected tensor from the downsampling path. During blending the smaller tensor is upsampled bilinearly to match the resolution of the skip-connected tensor. Then the two of them are blended either by summation or by concatenation followed by a convolution. Note that the upsampling path is very lean when compared to downsampling path. Each upsampling unit

usually contains only one or two 1×1 convolutions. This type of upsampling for semantic segmentation was first introduced by U-Net [42]. The main difference of our approach to U-Net is that U-Net has a symmetric downsampling and upsampling path. We propose an asymmetric model which is better suited for most dense prediction problems. The intuition behind this is that recognition task in the downsampling path is harder than the task of feature blending and resolution recovery in the upsampling path. Therefore, the downsampling path requires a heavy model with much larger capacity than the upsampling path. Furthermore, U-Net uses concatenation followed by two 3×3 convolutions in its upsampling unit and has one 2×2 convolution applied after each upsampling where the number of feature maps is reduced.

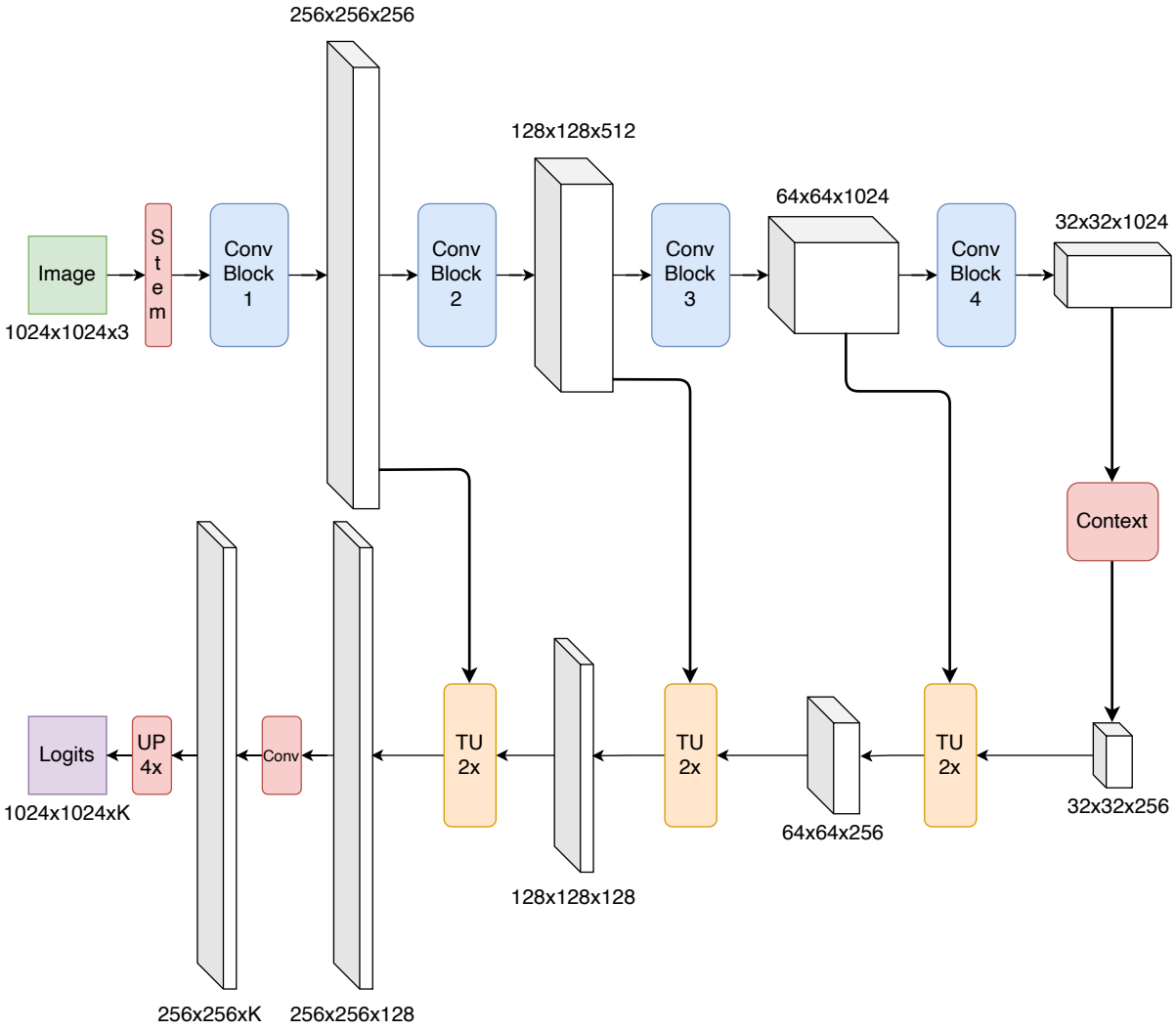

Figure 2.9: Semantic segmentation model obtained by utilizing ladder-style upsampling. The upper downsampling path is the same ImageNet model from Figure 2.3. The lower upsampling path recovers the lost resolution by blending the higher level features with the lower level features via skip connections to the intermediate feature tensors from the downsampling path. K denotes the number of classes. Tensor shapes are in $H \times W \times C$ format and correspond to the DenseNet121 model [20].

Chapter 3

Convolutional Scale Invariance for Semantic Segmentation

In this chapter we develop an approach towards introducing scale invariance to convolutional networks for semantic segmentation. While it is easy to design convolutional networks that are invariant to translation (cf. Chapter 2), there is no simple way to make them invariant to different appearance scales and rotation in the image.

We motivate our approach by considering semantic segmentation as classification of image windows centered at the particular pixel. This formulation resembles the localization task [43], which is concerned with finding objects in images. Localization similarly performs classification of image windows of different shape and scale. But in localization those image windows describe the bounding boxes around objects. However, the two tasks are trained in a different manner. Positive object localization windows have a well-defined spatial extent: they are tightly aligned around particular instances of the considered class. On the other hand, the class of a semantic segmentation window is exclusively determined by the kind of the object which projects to the central pixel of the window. Thus we see that the window size does not affect the correct semantic segmentation outcome. However, this opens the question of how large should these windows be. Unfortunately, there is no simple answer to this question. Featureless or ambiguous texture calls for large windows in order to allow the model to squeeze information from the context [44]. Small distinctive objects require small windows, since off-object pixels may provide misleading classification cues. This suggests that a pixel-level classifier is likely to perform better if supplied with a local image representation at multiple scales [30, 32, 45] and multiple levels of detail [46, 47].

We especially consider applications in intelligent transportation systems and robotics. We note that images acquired from vehicles and robots are quite different from images taken by humans. Images taken by humans always have a purpose: the photographer

wants something to be seen in the image. On the other hand, a vehicle-mounted camera operates independently from the pose of the objects in the scene: it simply acquires a fresh image each 40 milliseconds. Hence, the role of the context [44] in car-borne datasets [48, 49] will be different than in datasets acquired by humans [50]. In particular, objects in car-borne datasets (cars, pedestrians, riders, etc) are likely to be represented at a variety of scales due to forward camera motion. Not so in datasets taken by humans, where a majority of objects is found at particular scales determined by rules of artistic composition. This suggests that paying a special attention to object scale in car-borne imagery may bring a considerable performance gain.

One approach to address scale-related problems would be to perform a joint dense recovery of depth and semantic information [51, 52]. If the depth recovery is successful, the classification network gets an opportunity to leverage that information and improve performance. However, these methods have limited accuracy and require training with depth groundtruth. Another approach would be to couple semantic segmentation with reconstructed [53] or measured [54] 3D information. However, the pixel-level classifier may not be able to successfully exploit this information. Yet another approach would be to use the depth for presenting better object proposals [54, 55]. However, proposing instance locations in crowded scenes may be a harder task than classifying pixels.

In this chapter we present a novel technique for scale-invariant training and inference in stereoscopic semantic segmentation. This work was accepted for oral presentation at GCPR 2016 [1]. Unlike previous approaches, we address the scale-invariance directly, by leveraging the reconstructed depth information [56, 57] to disentangle the object appearance from the object scale. We realize this idea by introducing a novel scale selection layer into a deep network operating on the source image pyramid. The resulting scale-invariance substantially improves the segmentation performance with respect to the baseline.

3.1 Related work

Early semantic segmentation work was based on multi-scale hand-crafted filter banks [2, 28] with limited receptive fields (typically less than 50×50 pixels). Recent approaches [32, 45, 46, 58] leverage amazing power of GPU [30] and extraordinary capacity of deep convolutional architectures [6] to process pixel neighborhoods by ImageNet-grade classifiers. These classifiers typically possess millions of trainable parameters, while their receptive fields may exceed 200×200 pixels [16]. The capacity of these architectures is dimensioned for associating an unknown input image with one of 1000 diverse classes, while they typically see around million images during ImageNet pre-training plus billions of patches during semantic segmentation training. An architecture trained for ImageNet classifi-

cation can be transformed into a fully convolutional form by converting fully-connected layers into equivalent convolutional layers with the same weights [46, 59, 60]. The resulting fully convolutional network outputs a dense W/s×H/s×1000 multi-class heat map tensor, where W×H are input dimensions and s is the subsampling factor due to pooling. Now the number of outputs of the last layer can be redimensioned to whatever is the number of classes in the specific application and the network is ready to be fine-tuned for the semantic segmentation task.

Convolutional application of ImageNet architectures typically results in considerable downsampling of the output activations with respect to the input image. Some researches have countered this effect with trained upsampling [46] which may be reinforced by taking into account switches from the strided pooling layers [61, 62]. Other ways to achieve the same goal include interleaved pooling [59] and dilated convolutions [35, 46]. These approaches typically improve the baseline performance in the vicinity of the object borders due to more accurate upsampling of the semantic maps. We note that the system presented in our experiments does not feature any of these techniques, however it still succeeds to deliver competitive performance.

As emphasized in the introduction, presenting a pixel-level classifier with a variety of local image representations is likely to favor semantic segmentation performance. Previous researchers have devised two convolutional approaches to meet this idea: the skip architecture and the shared multi-class architecture. The shared multi-class architectures concatenate activations obtained by applying the common pixel-level classifier at multiple levels of the image pyramid [16, 30, 32, 45, 51]. The skip architectures concatenate activations from different levels of the deep convolutional hierarchy [46, 47]. Both architectures have their merits. The shared multi-scale architecture is able to associate the evaluated window with the training dataset at unseen scales. The skip architecture allows to model the object appearance [43] and the surrounding context at different levels of detail, which may be especially appropriate for small objects. Thus it appears that best results might be obtained by a combined approach which we call a multi-scale skip architecture. The combined approach concatenates pixel representations taken at different image scales and at different levels of the deep network. We note that this idea does not appear to have been addressed in the previous work.

Despite extremely large receptive fields, pixel-level classification may still fail to establish consistent activations in all cases. Most problems of this kind concern smooth parts of very large objects: for example, a pixel in the middle of a tram may get classified as a bus. A common approach to such problems is to require pairwise agreement between pixel-level labels, which leads to global optimization across the entire image. This requirement is often formulated as MAP inference in conditional random fields (CRF)

with unary, pairwise [31, 32, 58] and higher-order potentials [63]. Early methods allowed binary potentials exclusively between neighboring pixels, however, this requirement has later been relaxed by defining binary potentials as linear combinations of Gaussian kernels. In this case, the message passing step in approximate mean field inference can be expressed as a convolution with a truncated Gaussian kernel in the feature space [31, 58]. Recent state of the art approaches couple the CRF training and inference in custom convolutional [32] and recurrent [63] deep neural networks. Experiments from this chapter feature a separately trained CRF with Gaussian potentials while our future work shall include joint CRF training [32].

We now review the details of the previous research which is most closely related to our contributions. Banica et al. [54] exploit the depth sensed by RGBD sensors to improve the quality of region proposals and the subsequent region-level classification on indoor datasets. Chen et al [45] propose a scale attention mechanism to combine classification scores at different scales. This results in soft pooling of the classification scores across different classes. Ladicky and Shi [51] propose to train binary pixel-level classifiers which detect semantic labels at some canonical depth by exploiting the depth groundtruth obtained by LIDAR. Their inference jointly predicts semantic segmentation and depth by processing multiple levels of the monocular image pyramid. Unlike all previous approaches, our technique achieves efficient classification and training due to scale-invariant image representation recovered by exploiting reconstructed depth. Unlike [45], we perform hard scale selection at the representation level. Unlike [51], we exploit the reconstructed instead of the groundtruth depth.

3.2 Fully convolutional architecture with scale selection

We integrate the proposed technique into an end-to-end trained fully convolutional architecture illustrated in Fig. 3.1. The proposed architecture independently feeds images from an N-level image pyramid into the shared feature extraction network. Features from the lower levels of the pyramid are upsampled to match the resolution of features from the original image. We forward the recovered multi-scale representation to the pixel-wise scale selection multiplexer. The multiplexer is responsible for establishing a scale-invariant image representation in accordance with the reconstructed depth at the particular pixel. The back-end classifier scores the scale-invariant features with a multi-class classification model. The resulting score maps are finally converted into the per-pixel distribution over classes by a conventional softmax layer.

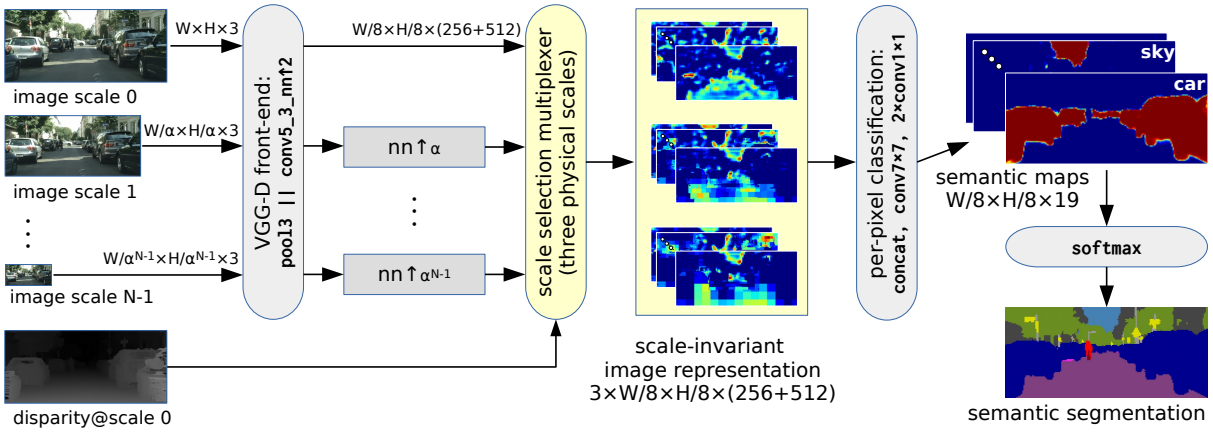

Figure 3.1: A convolutional architecture with scale-invariant representation.

3.2.1 Input pyramid and depth reconstruction

The left image of the input stereo pair is iteratively subsampled to produce a multi-scale pyramid representation. The first pyramid level contains the left input image. Each successive level is obtained by subsampling its predecessor with the factor α. If the original image resolution is W×H, the resolution of the l-th level is $W/\alpha^l \times H/\alpha^l$, $l \in [0..N\text{-}1]$ (α=1.3, N=8). We reconstruct the depth by employing a deep correspondence metric [57] and the SGM [56] smoothness prior. The resolution of the disparity image is W×H.

3.2.2 Single scale feature extraction

Our single scale feature extraction architecture is based on the feature extraction front-end of the 16-level deep VGG-D network [16] up to the `relu5_3` layer (13 weight layers total). In order to improve the training, we introduce a batch normalization layer [10] before each non-linearity in the 5th group: `relu5_1`, `relu5_2` and `relu5_3`. This modification helps the fine-tuning by increasing the flow of the gradients during backprop. Subsequently we perform a 2×2 nearest neighbor upsampling of `relu5_3` features and concatenate them with `pool3` features in the spirit of skip architectures [46, 47] (adding `pool4` features did not result in significant benefits). In comparison with `relu5_3`, the representation from `pool3` has a 2×2 higher resolution and a smaller receptive field (40×40 vs 185×185). We hypothesize that this saves some network capacity because it relieves the network from propagating small objects through all 13 convolutional and 4 pooling layers.

The described feature extraction network is independently applied at all levels of the pyramid, in the spirit of the shared multi-class architectures [16, 30, 32, 45, 51]. Subsequently, we upsample the representations of pyramid levels 1 to N-1 in order to revert the effects of subsampling and to restore a common resolution across the representations at all scales. We perform the upsampling by a nearest neighbor algorithm in order to

preserve the sparsity of the features. After upsampling, all N feature tensors have the resolution W/8×H/8×(512+256). The 8×8 subsampling is due to three pooling levels with stride 2. Features from `relu5_3` have 512 dimensions while features from `pool3` have 256 dimensions.

The described procedure produces a multi-scale convolutional representation of the input image. A straight-forward approach to exploit this representation would be to concatenate the features at all N scales. However, that would imply a huge computational complexity which would make training and inference infeasible. One could also perform such procedure at some subset of scales. However, that would require a costly validation to choose the subset, while providing less information to the back-end classification network. Consequently, we proceed towards achieving scale-invariance as the main contribution of our work.

3.2.3 Scale selection multiplexer

The responsibility of the scale selection multiplexer is to represent each image pixel with scale-invariant convolutional features extracted at exactly M=3 out of N levels of the pyramid. The scale invariance is achieved by choosing the pyramid levels in which the apparent size of the reference metric scales are closest to the receptive field of our features.

In order to explain the details, we first establish the notation. We denote the image pixels as p_i, the corresponding disparities as d_i, the stereo baseline as b, and the reconstructed depths as Z_i. We then denote the width of the receptive field for our largest features (`conv5_3`) as $w_{rf} = 185$, the metric width of its back-projection at distance Z_i as W_i and the three reference metric scales in meters as $W_R = \{1, 4, 7\}$. Finally, we define s_{mi} as the ratio between the m-th reference metric scale W_{Rm} and the back projection W_i of the receptive field:

$$s_{mi} = \frac{W_{Rm}}{W_i} = \frac{W_{Rm}}{\frac{b}{d_i} w_{rf}} = \frac{d_i \cdot W_{Rm}}{b \cdot w_{rf}} \,. \tag{3.1}$$

The ratio s_{mi} represents the exact image scaling factor by which we should downsample the original image to attain the reference scale m at pixel i. Now we are able to choose the representation from the pyramid level l_{mi} which has a downsampling factor closest to the true factor s_{mi}:

$$\hat{l}_{mi} = \underset{l}{\operatorname{argmin}} \left| \alpha^l - s_{mi} \right|, \quad l \in \{0, 1, ..., N-1\} \,. \tag{3.2}$$

The multiplexer determines the routing information at pixel p_i, by mapping each of the M reference scales to the corresponding pyramid level l_{mi}. We illustrate the recovered l_{mi}

in Fig. 3.2 by color coding the computed pyramid levels at three reference metric scales. Note that in case when $s_{mi} < 1$ we simply always choose the first pyramid level ($l = 0$). We

Figure 3.2: Visualization of the scale selection switches. From left to right: original image, switches for the three reference metric scales of 1m, 4m and 7m.

have not experimented with upsampled levels of the pyramid mostly because of memory limitations. The output of the multiplexer is a scale-invariant image representation which is stored in M feature tensors of the dimension W/8×H/8×(512+256).

3.2.4 Back-end classifier

The scale-invariant feature tensors are concatenated and passed on to the classification subnetwork which consists of one 7×7 and two 1×1 convolution+ReLU layers. The former two layers have 1024 maps and batch normalization before non-linearities. The last 1×1 convolutional layer is configured in a way that the number of feature maps corresponds to the number of classes. The resulting class scores are passed to the pixel-wise softmax layer to obtain the distribution across classes for each pixel.

3.3 A novel semantic segmentation dataset

We contribute a novel RGB-D dataset for semantic segmentation of driving scenes. The dataset contains groundtruth semantic segmentations for 445 hand-picked images from the KITTI dataset [48]. We start from 146 images annotated by German Ros from UAB Barcelona [2], improve their annotation accuracy and contribute another 299 images. The annotations feature high quality pixel-level polygonal approximations into 11 semantic classes: building, vegetation, sky, road, fence, pole, sidewalk, sign, car, pedestrian, bicyclist. The RGB images of size 1241×376 are provided with their corresponding depth images obtained with stereoscopic reconstruction. Some examples of annotations are depth images are given in Figure 3.3. The KITTI dataset provides a large collection of traffic videos with LIDAR reconstruction groundtruth. We use the LIDAR as groundtruth and utilize metric learning to train the convolutions network on the patches of size 11×11. The embeddings from convolutional network are used as an input to the Semi-global matching [56] algorithm where finally the stereoscopic depth image is produced. The combined dataset with 399 training and 46 test images is freely available for academic research [64, 65].

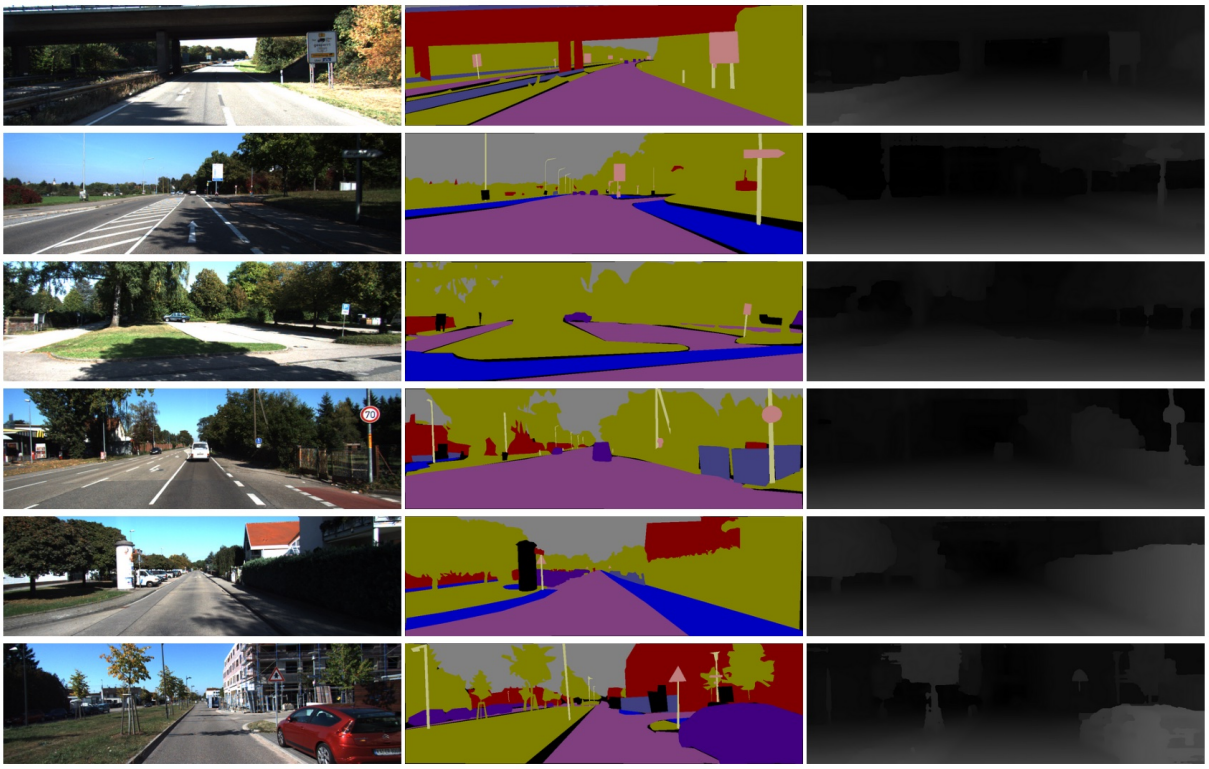

Figure 3.3: Some examples from KITTI semantic segmentation dataset. Camera images are taken from the KITTI dataset. We contribute the semantic segmentation annotations (first column)(middle column) and stereoscopic depth images (third column).

3.4 Experiments

We evaluate our method on two different semantic segmentation datasets containing outdoor traffic scenes: Cityscapes [49] and KITTI [48]. The Cityscapes dataset [49] has 19 classes recorded in 50 cities during several months. The dataset features good and medium weather conditions, large number of dynamic objects, varying scene layout and varying background. It consists of 5000 images with fine annotations and 20000 images with coarse annotations (we use only the fine annotations). The resolution of the images is 2048×1024. The dataset includes the stereo views which we use to reconstruct the depth.

We train our networks using Adam SGD [66] and batch normalization [10] without learnable parameters. Due to memory limitations we only have one image in a batch. The input images are zero-centered and normalized. We initialize the learning rate to 10^{-5}, decrease it to $0.5 \cdot 10^{-5}$ after 2nd epoch and again to 10^{-6} after 10th epoch. Before each epoch, the training set is shuffled to eliminate the bias. The first 13 convolutional layers are initialized from VGG-D [16] pretrained on ImageNet and fine tuned during training. All other layers are randomly initialized. In all experiments we train our networks for 15 epochs on Cityscapes dataset and 30 epochs on KITTI dataset. We use the softmax

cross-entropy loss which is summed up over all the pixels in a batch. In both datasets the frequency of pixel labels is highly unevenly distributed. We therefore perform class balancing by weighting each pixel loss with the true class weight factor w_c. This factor can be determined from class frequencies in the training set as follows: $w_c = \min(10^3, p(c)^{-1})$, where $p(c)$ is the frequency of pixels from the class c in the batch. In all experiments the three reference metric scales were set to equidistant values $W_R = \{1, 4, 7\}$ which were not cross-validated but had a good coverage over the input pixels (cf. Fig. 3.2). A Torch implementation of this procedure is freely available for academic research*.

In order to alleviate the downsampling effects and improve consistency, we postprocess the semantic map with the fully-connected CRF [31]. The negative logarithms of probability distributions across classes are used as unary potentials, while the pairwise potentials are based on a linear combination of two Gaussian kernels [31]. We fix the number of mean field iterations to 10. The smoothness kernel parameters are fixed at $w^{(2)} = 3, \theta_\gamma = 3$, while a coarse grid search on 200 Cityscapes images is performed to optimize $w^{(1)} \in \{5, 10\}, \theta_\alpha \in \{50, 60, 70, 80, 90\}, \theta_\beta \in \{3, 4, 5, 6, 7, 8, 9\}$.

The segmentation performance is measured by the intersection-over-union (IoU) score [67] and the pixel accuracy [46]. We first evaluate the performance of two single scale networks which are obtained by eliminating the scale multiplexer layer and applying our network to full resolution images (cf. Fig. 3.1). This network is referred to as Single3+5. Furthermore, we also have the Single5 network which is just Single3+5 without the representation from `pool3`. The label ScaleInvariant shall refer to the full architecture visualized in Fig. 3.1. The label FixedScales refers to the similar architecture with fixed multi-scale representation obtained by concatenating the pyramid levels 0, 3 and 7.

First, we show results on the Cityscapes dataset. We downsample original images and train on smaller resolution (1504x672) due to memory limitations. Table 3.1 shows the results on the validation and test sets. We can observe that our scale invariant network improves over the single scale approach across all classes in Table 3.1. We can likewise notice that the concatenation from pool3 is important as Single3+5 produces

*`https://github.com/ivankreso/scale-invariant-cnn`

Table 3.1: Individual class results on the Cityscapes validation and test sets (IoU scores). Last row represents a fraction by which each class is represented in the training set.

method	road	sidewalk	building	wall	fence	pole	traffic light	traffic sign	vegetation	terrain	sky	person	rider	car	truck	bus	train	motorcycle	bicycle	mean
Validation set																				
Single5	93.0	64.6	80.6	28.7	37.5	30.4	38.5	43.6	81.9	42.3	81.5	57.2	37.3	83.4	36.0	43.9	37.1	38.6	58.8	53.4
Single3+5	93.8	67.9	83.2	30.1	37.4	37.8	45.3	55.5	86.3	49.3	85.9	63.8	40.9	87.0	32.7	50.5	31.7	37.1	63.5	56.8
FixedScales	94.6	70.9	85.5	42.5	42.4	39.6	46.2	55.3	86.5	50.3	**86.4**	64.4	45.4	88.3	47.2	60.8	54.0	43.1	63.9	61.4
ScaleInvariant	**95.1**	**72.4**	**86.6**	**45.3**	**47.3**	**42.0**	**50.6**	**56.3**	**87.2**	**52.7**	86.1	**69.8**	**51.1**	**89.0**	**55.3**	**70.8**	**54.0**	**46.8**	**64.1**	**64.4**
Test set																				
ScaleInvariant	95.3	73.5	86.4	36.8	42.7	45.5	56.6	57.8	89.5	63.6	90.3	73.5	53.1	**90.3**	30.8	48.2	39.6	52.2	62.7	62.5
ScaleInvariant+CRF	**96.3**	**76.8**	**88.8**	**40.0**	**45.4**	**50.1**	**63.3**	**69.6**	**90.6**	**67.1**	**92.2**	**77.6**	**55.9**	90.1	**39.2**	**51.3**	**44.4**	**54.4**	**66.1**	**66.3**
% in train	33.5	6.3	24.0	0.7	1.0	1.4	0.2	0.6	16.9	1.2	4.0	1.4	0.2	7.6	0.3	0.3	0.3	0.1	0.5	

better results then Single3 and that improvement is larger for smaller classes like poles, traffic signs and traffic lights. That supports our hypothesis that the representation from `pool3` helps to better handle smaller objects. Furthermore, the scale invariant network achieves significant improvement over multi-scale network with fixed image scale levels (FixedScales). This agrees with our hypothesis that the proposed scale selection approach should help the network to learn a better representation. The table also shows results on the test set (our online submission is entitled Scale invariant CNN + CRF).

The last row in Table 3.1 represents a proportion by which each class is represented in the training set. We can notice that the greatest contribution of our approach is achieved for classes which represent smaller objects or objects that we see less often like buses, trains, trucks, walls etc. This effect is illustrated in Fig. 3.4 where we plot the improvement of the IoU metric with respect to the training set proportion for each class. Likewise we achieve improvement in pixel accuracy from 90.1% (Single3+5) to 91.9% (ScaleInvariant).

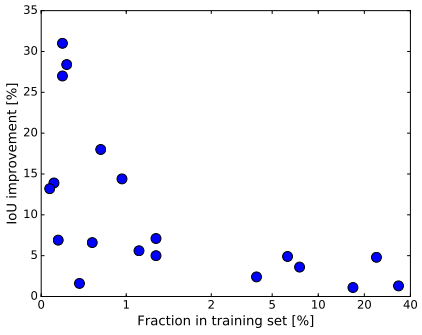

Figure 3.4: Improvement of the IoU metric between Single3+5 and ScaleInvariant architecture with respect to the proportion inside training set for each class.

Fig. 3.5 shows examples where the scale-invariant network produces better results. The improvement for big objects is clearly substantial. We often observe that the scale-invariant network can differentiate between road and sidewalk and person and rider, especially when they assume a rare appearance as in the last row with cobbled road which is easily mistaken for sidewalk.

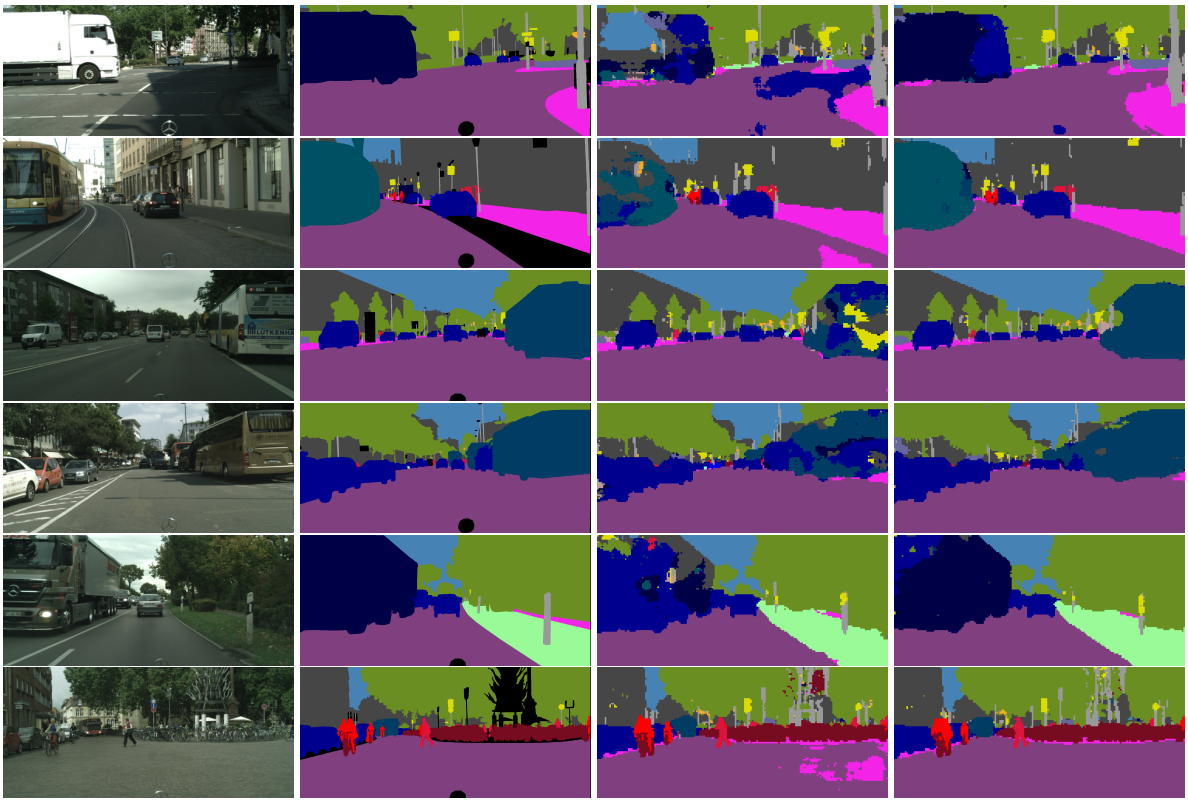

Figure 3.5: Examples where scale invariance helps the most. From left to right: input, groundtruth, baseline segmentation (Single3+5), scale invariant segmentation.

Table 3.2: Individual class results on the KITTI test set (IoU scores).

method	sky	building	road	sidewalk	fence	vegetation	pole	car	sign	pedestrian	cyclist	mean
Single3+5	82.1	82.1	89.5	71.8	37.1	**80.7**	27.4	79.7	32.6	41.9	**14.3**	58.1
ScaleInvariant	**84.1**	**83.9**	**91.4**	**73.1**	**43.2**	79.2	**33.6**	**82.0**	**44.7**	**57.3**	12.8	**62.3**

Table 3.2 shows results on the KITTI test set. We notice a significant improvement in mean IoU class metric, which is, however, smaller than on Cityscapes. The main reason is that KITTI has much less smaller classes (only pole, sign and cyclist). Furthermore, it is a much smaller dataset which explains why the performance is so low on classes like cyclist and pole. Here we again report an improvement in pixel accuracy from 87.63 (Single3+5) to 88.57 (ScaleInvariant).

3.5 Discussion

We have presented a novel technique for improving semantic segmentation performance. We use the reconstructed depth as a guide to produce a scale-invariant representation in which the appearance is decoupled from the scale. This precludes the necessity to recognize objects at all possible scales and allows for an efficient use of the classifier

capacity and the training data. This trait is especially important for navigation datasets which contain objects at a great variety of scales and do not exhibit the photographer bias.

We have integrated the proposed technique into an end-to-end trainable fully convolutional architecture which extracts features by a multi-scale skip network. The extracted features are fed to the novel multiplexing layer which carries out dense scale selection at the pixel level and produces a scale-invariant representation which is scored by the back-end classification network.

We have performed experiments on the novel Cityscapes dataset. Our results are very close to the state-of-the-art, despite the fact that we have trained our network on reduced resolution. We also report experiments on the KITTI dataset where we have densely annotated 299 new images, improved 146 already available annotations and release the union of the two datasets to the community. The proposed scale selection approach has consistently contributed substantial increases in segmentation performance. The results show that deep neural networks are extremely powerful classification models, however, they are still unable to learn geometric transformation better than humans.

Chapter 4

Efficient Ladder-style DenseNets for Semantic Segmentation of Large Images

In this chapter we develop and present an efficient convolutional architecture for semantic segmentation of large images based on ladder-style upsampling. The backbone of the architecture is based on DenseNet [20]. We show that DenseNet is an efficient model for dense prediction when combined with ladder-style upsampling and due to its heavy feature reuse, very suitable for semantic segmentation.

Much recent attention has been directed towards residual models (also known as ResNets) [18, 21] in which each processing step is expressed as a sum between a compound non-linear unit and its input. This introduces an auxiliary information path which allows a direct gradient propagation across the layers, similarly to the flow of the state vector across LSTM cells. However, in contrast to the great depth of residual models, it appears that most of the training occurs along relatively short paths [68]. Hence, today we believe that residual models act as an exponentially large ensemble of moderately deep sub-models. Recent approaches replicate and exceed the success of residual models by introducing skip-connections across layers. Our work is based on densely connected models (also known as DenseNets) [19]. As described in Section 2.6, convolutional units of densely connected blocks operate on concatenations of all previous features at the current resolution. This encourages feature sharing and discourages overfitting [19], while favouring the gradient flow towards early layers as in much better known ResNet models. Authors of the DenseNet concept have succeed to match the Imagenet accuracy of residual models with three times more parameters. Our semantic segmentation experiments show that DenseNet-based models outperform their counterparts based on ResNets [21] and more recent dual path networks [69]. Besides outstanding generalization, DenseNets carry a

great memory-saving potential due to extensive feature reuse. However, this potential is not easily materialized due to inefficient activation caching during automatic differentiation. We show that this can be effectively alleviated by extensive gradient checkpointing [70] which leads to five-fold reduction of memory requirements.

Most deep models for semantic segmentation must decrease the spatial resolution of deep layers in order to enable training under strict GPU memory limitations. Subsequently, deep features are carefully upsampled in order to generate correct predictions at semantic borders and small objects. Most previous work reduces the subsampling by dilated filtering [36, 37, 38, 39, 40]. We take another approach, where deep features are upsampled by exploiting activations from earlier layers. Different than previous such methods [71, 72, 73, 74, 75], we conjecture that upsampling requires much less capacity than the downsampling path. Consequently, we propose a minimalistic upsampling datapath which is very well suited for efficient processing of large images.

We present an effective lightweight architecture for semantic segmentation of large images, based on DenseNet features and ladder-style [76] upsampling. We propose several improvements with respect to our previous work [4], which lead to better accuracy and faster execution while using less memory and fewer parameters. Our consolidated contribution is three-fold. First, we increase computational efficiency by subsampling the representation in the middle of a DenseNet block. Second, we propose an efficient ladder-style upsampling datapath which requires less memory and achieves a better $\overline{\text{IoU}}$/FLOP trade-off than related previous work [71, 72]. Third, we unlock the potential of our method for memory-efficient training with a novel checkpointing approach. Our method now requires less than half memory than ResNet with in-place activated batchnorm [39]. These contributions strike an excellent balance between prediction accuracy and model complexity. Experiments on Cityscapes, CamVid, ROB 2018 and Pascal VOC 2012 demonstrate state-of-the-art recognition performance and competitive inference speed.

4.1 Motivation for ladder-style upsampling

Early convolutional models for semantic segmentation had only a few pooling layers and were trained from scratch [30]. Later work built on feature extractors pre-trained on ImageNet [16, 18, 19], which produce 32× subsampled representations. This improved generalization [77] and reduced the training footprint due to smaller resolution of feature tensors. Early upsampling approaches leveraged trained filters [78] and cached max-pool switches [79]. Many recent approaches modify some strided layers to produce non-strided output while doubling the dilation factor of all subsequent convolutions [37, 59]. This decreases the extent of subsampling while preserving pre-trained semantics of the feature

extractor.

In principle, dilated filtering can completely recover the resolution without compromising pre-trained parameters. However, we believe that dilated filtering should be used sparingly [80] or completely avoided [4, 33] due to following two shortcomings. First, dilated filtering substantially increases the training footprint and inference complexity, and thus hinders single-GPU training and real-time inference. Practical implementations alleviate this by recovering only up to the last two subsamplings, which allows subsequent inference at 8× subsampled resolution [36, 37]. Second, dilated filtering treats semantic segmentation exactly as if it were ImageNet classification: each pixel is classified independently from its neighbours. This does not feel right since we know that neighboring pixels are highly correlated in practice.

We therefore prefer to keep the downsampling and restore the resolution by blending semantics of deep features with location accuracy of the earlier layers [46]. This encourages the deep layers to discard location information and focus on abstract image properties [76]. Practical realizations avoid high-dimensional features at output resolution [46] by ladder-style upsampling [33, 71, 76].

Note that our upsampling approach is different from the symmetric encoder-decoder approaches [71, 72, 79], where the upsampling datapath mirrors the structure of the downsampling datapath. These methods are unable to process large images or deliver high execution speed due to excessive upsampling complexity.

We briefly review other related upsampling approaches. Ghiasi et al [73] blend predictions (instead of blending features) by prefering the deeper layer in the middle of the object, while favouring the earlier layer near the object boundary. Pohlen et al [81] propose a two-stream residual architecture where one stream is always at the full resolution, while the other stream is first subsampled and subsequently upsampled by blending with the first stream. Lin et al [74] perform the blending by a sub-model comprised of 8 convolutional and several other layers in each upsampling step. Islam et al [82] blend upsampled predictions with two layers from the downsampling datapath. This results in 4 convolutions and one elementwise multiplication in each upsampling step. Peng et al [75] blend predictions produced by convolutions with very large kernels. The blending is performed by one 3×3 deconvolution, two 3×3 convolutions, and one addition in each upsampling step.

Contrary to previous approaches, we conjecture that it is much more difficult to recognize semantic classes than to delineate semantic borders. We therefore argue for an asymetric encoder-decoder approach consisting of a powerful backbone and efficient upsampling. Each upsampling step has only one 3×3 convolution whereby the number of feature maps is much lower than in the corresponding layers of the downsampling path [4].

To the best of our knowledge, such organization has not been previously used for semantic segmentation, although there exists related work in object detection [33], and instance-level segmentation [83]. Note that our lateral connections differ from [73, 75, 82], since they blend predictions, while we blend features. Blending features improves the modelling power, but is more computationally demanding. We can afford to blend features due to efficient upsampling and gradient checkpointing which will be explained later.

4.2 Motivation for spatial pyramid pooling

Pretrained convolutional representations are not directly applicable to large images due to insufficient receptive range. This issue typically shows up in large indistinctive regions which are common in urban scenes. These regions are often projected from nearby objects, which makes them very important for high-level tasks (as exemplified by circumstances of the first fatal incident of a level-2 autopilot). Some approaches address this issue with dilated convolutions [37, 80], however sparse sampling may hurt the accuracy due to aliasing. The receptive range can also be enlarged by increasing the model depth [82]. However, the added capacity may result in overfitting. Correlation between distant parts of the scene can also be modelled with long-range connections [32, 84]. However, these may be unsuitable due to large capacity and computational complexity. A better ratio between receptive range and complexity is achieved with spatial pyramid pooling (SPP) [36, 85, 86] which augments the features with their spatial pools over rectangular regions of varying size.

4.3 Motivation for densely connected segmentation models

To the best of our knowledge, there are only two previous works on DenseNet-based semantic segmentation [40, 72]. However, these approaches fail to position DenseNet as the backbone with the largest potential for memory-efficient feature extraction. This potential is caused by a specific design which encourages inter-layer sharing [19] instead of forwarding features across the layers. Unfortunately, automatic differentiation is unable to exploit this potential due to concatenation, batchnorm and projection layers. Consequently, straightforward DenseNet implementations actually require a little bit more memory than their residual counterparts [4]. Here we show that the problem can be resolved with checkpointing [19, 70]. In particular, we propose to recompute all batchnorm and projection activations during backprop. We show that dramatic improvements in memory footprint can be obtained by applying this idea both to the DenseNet-based

downsampling datapath and to the ladder-style upsampling datapath. Previous work on checkpointed segmentation models considered only residual models [39], and achieved only two-fold memory reduction. On the other hand, our custom checkpointing scheme is specifically crafted for our architecture, and is therefore able to achieve up to six-fold memory reduction with respect to the baseline.

4.4 The proposed architecture

We propose a semantic segmentation architecture aiming at high accuracy, high execution speed, and low memory footprint. The architecture is based on the DenseNet feature extractor [19], spatial pyramid pooling (SPP) [86] and minimalistic ladder-style blending [33, 76]. The proposed architecture consists of two datapaths which roughly correspond to two horizontal rails in Figure 4.1. The downsampling datapath includes a customized DenseNet feature extractor [19], and a lightweight spatial pyramid pooling module (SPP) [36]. The feature extractor transforms the input image into convolutional features **F** by gradually reducing spatial resolution, and increasing the number of feature maps (top rail in Fig. 4.1). The SPP module enriches **F** with context information, and produces context-aware features **C**. The upsampling datapath transforms **C** into high-resolution semantic predictions (bottom rail in Fig. 4.1) by exploiting fine details from the downsampling datapath.

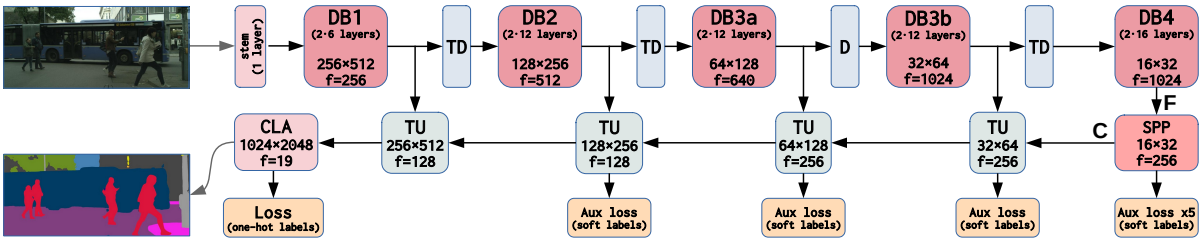

Figure 4.1: Diagram of the proposed segmentation architecture. Each processing block is annotated with dimensions of the resulting feature tensor in case of DenseNet-121 and Cityscapes input (f denotes the number of feature maps). We split the dense block DB3 in order to enlarge receptive range, improve speed and reduce the memory footprint. The transition-up (TU) blocks blend low-resolution semantic features with high-resolution low-level features. The classifer (CLA) projects features to 19 semantic maps and bilinearly upsamples them to the input resolution.

4.4.1 Feature extraction

Our feature extractor consists of densely connected blocks (DB) and transition layers (TD, D) (cf. Figure 4.1). Each DB is a concatenation of convolutional units, while each

convolutional unit operates on a concatenation of all preceding units and the DB input [19]. We customize the original DenseNet design by splitting DB3 into two fragments (DB3a and DB3b), and placing a strided average-pooling layer (D) in-between them. This enlarges the receptive field of all convolutions after DB3a, while decreasing their computational complexity. In comparison with dilated filtering [37], this approach trades-off spatial resolution (which we later restore with ladder-style blending) for improved execution speed and reduced memory footprint. Note that the resolution is restored without loss of accuracy with ladder-style upsampling, in the same way as in other parts of the architecture. We initialize DB3b filters with the original ImageNet-pretrained weights, although the novel pooling layer alters the features in a way that has not been seen during ImageNet pretraining. Despite this discrepancy, fine-tuning succeeds to recover and achieve competitive generalization. The feature extractor concludes by concatenating all DB4 units into the 64× subsampled representation **F**.

4.4.2 Spatial pyramid pooling

Spatial pyramid pooling captures wide context information [36, 85, 86] by augmenting **F** with average pools over spatial grids. We first project **F** to d/2 maps, where d denotes the

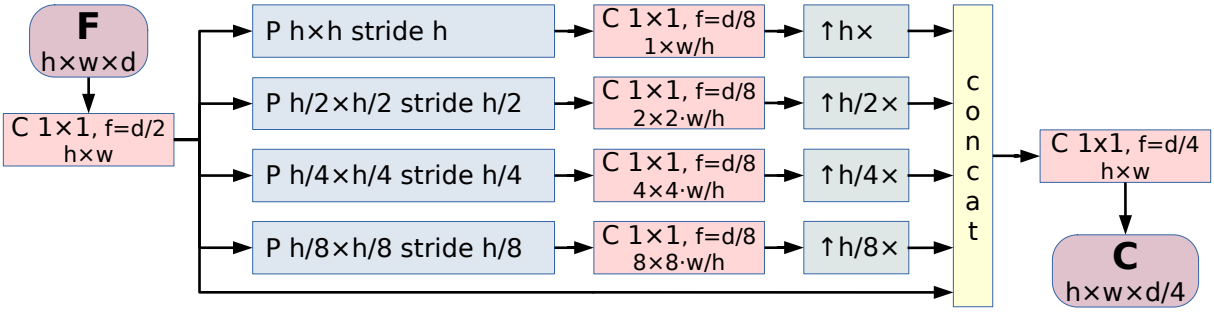

Figure 4.2: The proposed SPP module. P, C, and ↑ denote pooling, convolution and bilinear upsampling. We introduce two modifications with respect to [36]: i) we adapt the grid to the aspect ratio of input features **F**, and ii) we reduce the dimensionality throughout the module in order to decrease memory footprint and discourage overfitting.

dimensionality of features in **F**. The resulting tensor is then average-pooled over four grids with 1, 2, 4, and 8 rows. The number of grid columns is set in accordance with the image size so that all cells have a square shape. We project each pooled tensor to d/8 maps and then bilinearly upsample to the common resolution. We concatenate all results with the projected **F**, and finally blend with a 1×1×d/4 convolution. The resulting context-aware feature tensor **C** is h×w×d/4 where h=H/64, w=W/64, while H,W denotes the input resolution. If we use DenseNet-121 (d=1024), **C** has 48 times less dimensions than the input image.

4.4.3 Ladder-style upsampling

Ladder-style upsampling recovers fine details through a series of efficient transition-up (TU) blocks. Each TU block blends the preceding representation along the upsampling path ($\mathbf{U_{i-1}}$) with a skip-connection from the corresponding densely connected block (**DBx**). $\mathbf{U_{i-1}}$ has strong semantics while **DBx** has more spatial detail. The output $\mathbf{U_i}$ gets the best of both worlds. We first upsample $\mathbf{U_{i-1}}$ so that the two representations have the same resolution. Subsequently, we project **DBx** to a lower-dimensional space so that the two representations have the same number of feature maps. This balances the relative influence of the two datapaths and allows blending by simple summation. Subsequently, we reduce the dimensionality of the sum with 1×1 convolution, and blend the result with 3×3 convolution to deliver the output $\mathbf{U_i}$ as shown in Fig. 4.3. The described blending

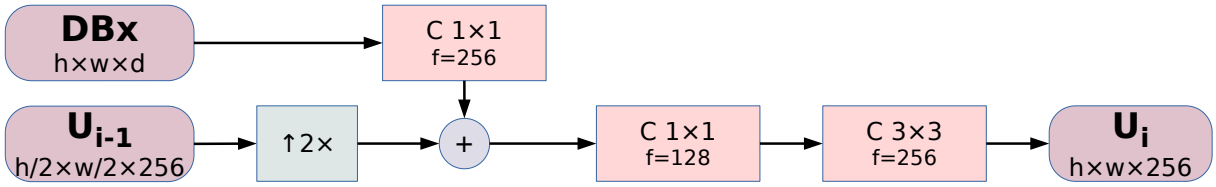

Figure 4.3: The proposed transition-up module (UP in Fig. 4.1). C, and ↑ denote convolution and bilinear upsampling. The number of feature maps in $\mathbf{U_{i-1}}$ and $\mathbf{U_i}$ varies according to Fig. 4.1. Minimalistic design ensures fast inference (only one 3×3 convolution), and low memory footprint (few convolutions, low feature dimensionality).

procedure is recursively applied along the upsampling datapath. The last transition-up block produces features at the resolution of the DenseNet stem. Finally, the CLA module recovers dense predictions by projection onto semantic classes and 4× bilinear upsampling. The proposed design has much fewer parameters than the downsampling datapath and therefore discourages overfitting as we show in the experiments.

4.5 Gradient checkpointing

Semantic segmentation requires a lot of caching during backprop, especially for large input resolutions. This may lead to difficulties due to strict limitations of GPU RAM. For example, it is well known that small training batches result in poor learning due to unstable batchnorm statistics. This problem can not be overcome by accumulating backward passes, and therefore hinders the accuracy of the model.

The extent of backprop-related caching can be reduced with gradient checkpointing [70]. The main idea is to instruct forward pass to cache only a carefully selected subset of all activations which are denoted as gradient checkpoints. The subgraph between two gradient checkpoints is denoted as a checkpointing segment. The backward pass iterates

over all checkpointing segments and processes them before continuing to the next segment as follows:

1. forward pass activations are recomputed starting from the checkpoint
2. the gradients are computed via the standard backward pass. The local cache is released as soon as a segment is processed.

We note that segment granularity affects space and time efficiency. Enlarging the checkpoint segments always reduces the memory footprint of the forward pass. However, the influence to backward pass memory requirements is non-trivial. Larger segments require more memory individually as they need to re-compute all required activations and store them in the local cache. At some point, we start to lose the gains obtained during forward pass. Our best heuristic was to checkpoint only outputs from 3×3 convolutions as they are the most compute-heavy operations. This heuristic is shown in Figure 4.4 on the example of computing the 8th convolutional unit. Note that in case when checkpointing is applied only the output of the unit is cached. The input tensors are composed of the 7 outputs from previous units and were therefore already cached. For this concrete example unit, this results in $1-32/(224*2+128*2+32) \approx 95\%$ less memory required (Sec. 2.7.3).

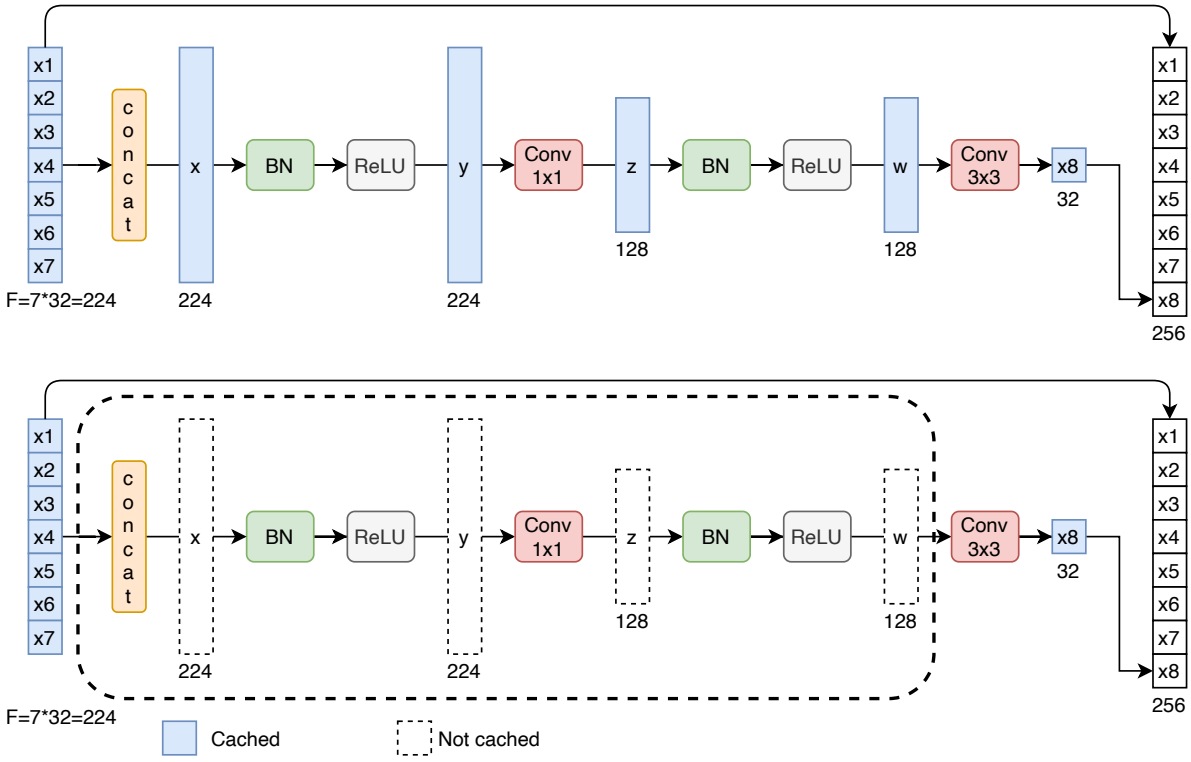

Figure 4.4: The proposed checkpointing strategy for DenseNet convolutional unit. All the cached tensors during forward pass are colored in blue. We show the forward pass of the 8th unit. The top row represents the baseline without gradient checkpointing while the bottom row shows the strategy where we checkpoint only the output from 3×3 convolution. Tensors drawn with dotted line are recomputed during the backward pass.

In other words, we propose to re-compute the stem, all projections, all batchnorms and all concatenations during the backward pass. Experiments show that this approach strikes a very good balance between maximum memory allocation in forward and backward passes.

The proposed checkpointing strategy is related to the previous approach [19] which puts a lot of effort into explicit management of shared storage. However, here we show that similar results can be obtained by relying on the standard PyTorch memory manager. We also show that custom backprop operations can be completely avoided by leveraging the standard PyTorch module `torch.utils.checkpoint`. Finally, we propose to achieve further memory gains by checkpointing outputs of 3×3 convolutions. This implies re-computing the stem, transition-down and transition-up blocks, and each DenseNet unit except the 3×3 convolution. To the best of our knowledge, this is the first account of applying extensive checkpointing for semantic segmentation.

4.6 Experiments

Most of our experiments target road-driving images since the corresponding applications require large input resolution (subsections 4.6.2, 4.6.3, and 4.6.4). For completeness, we also present results on photographic collections (4.6.5), and ablation experiments (4.6.6). Finally, we show that our method is extremely memory-efficient when used with checkpointing (4.6.7).

4.6.1 Training details and notation

We train our models with Adam optimizer [15] with the initial learning rate $4 \cdot 10^{-4}$. The learning rate is decreased after each epoch according to the cosine learning rate policy where we update the learning rate after each epoch e as follows:

$$\mathrm{lr} = \mathrm{lr}_{min} + \frac{\mathrm{lr}_{max} - \mathrm{lr}_{min}}{2} \cdot (1 + \cos(\frac{e}{e_{max}}\pi)) \tag{4.1}$$

We set lr_{max}=$4 \cdot 10^{-4}$, lr_{min}=10^{-7}, and $e_{max} = 300$. We divide the learning rate by 4 for all weights pre-trained on ImageNet belonging to the backbone downsampling model. Batch size is an important hyper-parameter of the optimization procedure. If we train with batch size 1, then the batchnorm statistics fit exactly the image we are training on. This hinders learning due to large covariate shift across different images [17]. We combat this by training on random crops with batch size 16. Before cropping, we apply a random flip and rescale the image with a random factor between 0.5 and 2. The crop size is set to 448, 512 or 768 depending on the resolution of the dataset. If a crop happens to be

larger than the rescaled image, then the undefined pixels are filled with the mean pixel. We train for 300 epochs unless otherwise stated. We employ multiple cross entropy losses along the upsampling path as shown in Figure 4.1. In order to promote efficient learning, auxiliary losses are formulated on the native k× subsampled resolution as cross-entropy between predictions $\hat{\mathbf{y}}$ and soft targets $\mathbf{y}^{\mathrm{k}}$:

$$\mathcal{L}_k^{\mathrm{aux}} = -\frac{k^2}{WH} \sum_{rc} \sum_{f} y_{rcf}^{\mathrm{k}} \log \hat{y}_{rcf} \,. \tag{4.2}$$

The soft targets correspond to the distribution of one-hot labels $\mathbf{y}^{\mathrm{OH}}$ in the corresponding k×k window:

$$\mathbf{y}_{rc}^{\mathrm{k}} = \frac{1}{k^2} \sum_{\Delta r \Delta c} \mathbf{y}_{kr+\Delta r, kc+\Delta c}^{\mathrm{OH}} \,. \tag{4.3}$$

We apply loss after each upsampling step, and to each of the four pooled tensors within the SPP module. The loss of the final predictions and the mean auxiliary loss are weighted with factors 0.6 and 0.4, respectively. These values were validated on the Cityscapes dataset. After the training, we recompute the batchnorm statistics as exact averages over the training set instead of decayed moving averages calculated during training. This practice slightly improves the model generalization.

We employ the following notation to describe our experiments throughout the section. LDN stands for Ladder DenseNet, the architecture proposed in Section 4.4. The symbol $d{\rightarrow}u$ denotes a model which downsamples the input resolution d times, and then uses ladder-style upsampling to produce predictions subsampled u times. For example, LDN121 64→4 denotes the model shown in Figure 4.1. Similarly, DDN and LDDN denote a dilated DenseNet, and a dilated DenseNet with ladder-style upsampling. The symbol $d{\downarrow}$ denotes a model which reduces the input resolution d times and has no upsampling path. MS denotes multi-scale evaluation on 5 scales (0.5, 0.75, 1, 1.5 and 2), and respective horizontal flips. $\overline{\mathrm{IoU}}$ and Cat. $\overline{\mathrm{IoU}}$ denote the standard mean IoU metric over classes and categories. The instance-level mean IoU ($\overline{\mathrm{iIoU}}$) metric [49] emphasizes contribution of pixels at small instances, and is therefore evaluated only on 8 object classes. The model size is expressed as the total number of parameters in millions (M). FLOP denotes the number of fused multiply-add operations required for inference on a single 1024×1024 (1MPx) image.

4.6.2 Cityscapes

Cityscapes has been recorded in 50 cities during daylight and fine weather [49]. It features 19 classes for test benchmark, many dynamic objects, and varying scene layout and background. The finely annotated part of the dataset consists of 2975 training, 500 validation,

and 1525 test images of 1024×2048 pixels.

Figure 4.5 shows the class incidence histogram for the Cityscapes dataset. Out of 29 total classes the 19 of them are used in training and evaluation. We can observe a large difference in the number of annotated pixels between target 19 classes due to some classes being less common in traffic scenes. The rare classes are: fence, wall, bus, truck, train, motorcycle and rider. Even though bicycle, traffic sign and traffic light classes have similar pixel coverage to bus and truck, they are a much smaller objects, resulting in them having significantly more individual object instances.

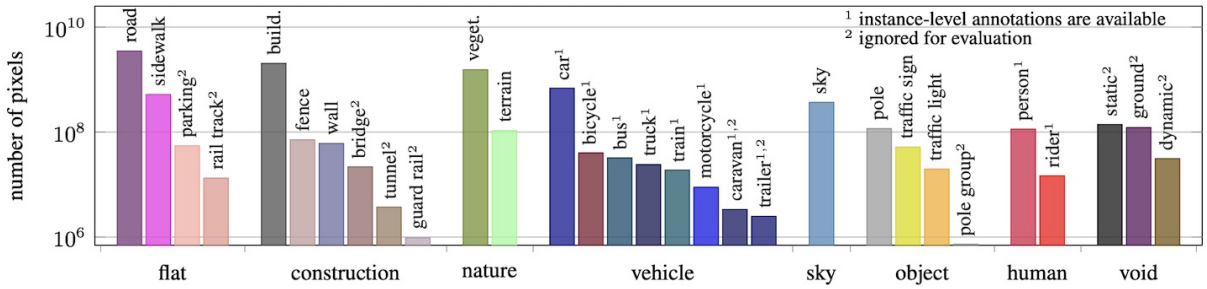

Figure 4.5: Class incidence for the Cityscapes dataset. Note that the incidence numbers are shown on the logarithmic scale. Classes denoted with superscript 2 are ignored in evaluation and training. We use the same class color coding in all our visualizations of the results. Figure reproduced from [49].

During our experiments we noticed a high variance in mIoU on the validation dataset when trying to reproduce the results with different PRNG seeds. Likewise, the same problem appears while evaluating the different model checkpoints during training. After comparing the individual class IoUs we concluded that the IoU on the validation dataset is very unstable for rare classes like train, bus, truck, motorcycle, wall and fence. Out of these rare classes the class *train* turned out to be the most problematic due to having the most unbalanced instance size representation in the dataset. One image contains a very large nearby instance of the train while the remaining few examples are all further away covering very small image area. In order to alleviate this problem for the train class we removed this problematic image from the validation set and moved it to the train set. Therefore all of our results on the validation set are measured on the 499 out of 500 original images. The prefix of the removed image is `frankfurt_000001_014565`.

Table 4.1 validates several popular backbones coupled with the same SPP and upsampling modules. Due to hardware constraints, here we train and evaluate on half resolution images, and use 448×448 crops. The first section presents the DN121 32↓ baseline. The second section presents our models with ladder-style upsampling. The LDN121 64→4 model outperforms the baseline for 10 percentage points (pp) of $\overline{\text{IoU}}$ improvement. Most of this improvement occurs on very small objects (due to blending with high-resolution

Table 4.1: Validation of backbone and upsampling architectures on Cityscapes val. Both training and evaluation images were resized to 1024×512.

Method	Class IoU	Class iIoU	Cat. IoU	Model size	FLOP 1MPx
DN121 32↓	66.2	46.7	78.3	8.2M	56.1G
LDN121 64→4	75.3	54.8	88.1	9.5M	66.5G
LDN121 32→4	**76.6**	57.5	88.6	9.0M	75.4G
LDN169 32→4	75.8	55.5	88.4	15.6M	88.8G
LDN121 32→2	**77.5**	**58.9**	89.3	9.4M	154.5G
ResNet18 32→4	70.9	49.7	86.7	13.3M	55.7G
ResNet101 32→4	73.7	54.3	87.8	45.9M	186.7G
ResNet50 32→4	73.9	54.2	87.8	26.9M	109.0G
DPN68 32→4	74.0	53.0	87.8	13.7M	59.0G
DDN-121 8↓	72.5	52.5	85.5	8.2M	147.8G
LDDN-121 8→4	75.5	55.3	88.3	8.6M	174.8G
LDDN-121 16→4	75.8	55.9	88.4	8.9M	87.0G

features) and very large objects (due to enlarged spatial context caused by increased subsampling, 64 vs 32). Note that LDN121 32→4 slightly outperforms LDN121 64→4 at this resolution due to better accuracy at semantic borders. However, the situation will be opposite in full resolution images due to larger objects (which require a larger receptive field) and off-by-one-pixel annotation errors. The LDN169 32→4 model features a stronger backbone, but obtains a slight deterioration (0.8pp) with respect to LDN121 32→4. We conclude that half resolution images do not contain enough training pixels to support the capacity of DenseNet-169. The LDN121 32→2 model shows that further upsampling doubles the computational complexity while bringing only a slight $\overline{\text{IoU}}$ improvement. The third section demonstrates that residual and DPN backbones achieve worse generalization than their DenseNet counterparts.

The last section presents the models which avoid the resolution loss by dilated convolutions. The DDN-121 8↓ model removes the strided pooling layers before the DenseNet blocks DB3 and DB4, and introduces dilation in DB3 (rate=2) and DB4 (rate=4). The SPP output is now 8× downsampled. From there we produce logits and finally restore the input resolution with bilinear upsampling. The LDDN-121 8→4 model continues with one step of ladder-style upsampling to obtain 4× downsampled predictions as in previous LDN experiments. We observe a 3pp $\overline{\text{IoU}}$ improvement due to ladder-style upsampling. The LDDN-121 16→4 model dilates only the last dense block and performs two steps of ladder-style upsampling. We observe a marginal improvement which, however, still comes short of LDN121 32→4 from Table 4.1. Training the DDN-121 4↓ model was infeasible due to huge computational requirements when the last three blocks operate on 4× sub-

sampled resolution. A comparison of computational complexity reveals that the dilated LDDN-121 8→4 model has almost 3× more FLOPs than LDN models with similar $\overline{\text{IoU}}$ performance. Finally, our memory consumption measurements show that LDDN-121 8→4 consumes around 2× more GPU memory than LDN121 32→4. We conclude that dilated models achieve a worse generalization than their LDN counterparts while requiring more computational power.

Table 4.2 shows experiments on full Cityscapes val images where we train on 768×768 crops. We obtain the most interesting results with the LDN121 64→4 model presented in Figure 4.1: 79% $\overline{\text{IoU}}$ with a single forward pass and 80.3% with multi-scale (MS) inference. Models with stronger backbones (DenseNet-169, DenseNet-161) validate only slightly better. We explain that by insufficient training data and we expect that successful models need less capacity for Cityscapes than for a harder task of discriminating ImageNet classes.

Table 4.2: Validation of various design options on full-resolution Cityscapes val.

Method	Class $\overline{\text{IoU}}$	Cat. $\overline{\text{IoU}}$	Model size	FLOP 1MPx
LDN121 32→4	78.4	90.0	9.0M	75.4G
LDN121 64→4	**79.0**	90.3	9.5M	66.5G
LDN121 128→4	78.4	90.1	9.9M	66.2G
LDN161 64→4	79.1	90.2	30.0M	138.7G
LDN121 64→4 MS	**80.3**	90.6	9.5M	536.2G

Table 4.3 compares our models with the state-of-the-art on validation and test sets. Our models generalize better than or equal to all previous approaches at the publication time of [3], while being much more efficient. In particular, we are the first to reach 80% $\overline{\text{IoU}}$ on Cityscapes test with only 66.5 GFLOP per Mpx. Table 4.4 presents a comparison with efficient approaches. It reports normalized processing time according to GPU conversion factors proposed in [77]. Label dws denotes depthwise separable convolutions in the upsampling path (cf. Table 4.9). Figure 4.6 plots the best performing models from Tables 4.3 and 4.4 in ($\overline{\text{IoU}}$,TFLOP) coordinates. The figure clearly shows that our models achieve the best trade-off between accuracy and computational complexity.

Table 4.3: Comparison with the state-of-the-art when training only on Cityscapes fine. FLOP metrics marked with '†' include only the backbone.

Method	Backbone	$\overline{\text{IoU}}$ Val	$\overline{\text{IoU}}$ Test	Tflop@1Mpx single scale
LKM [75]	rn50 d32↓	77.4	76.9	0.110†
TuSimple [87]	rn101 d8↓	76.4	77.6	0.720†
SAC-multiple [88]	rn101 d8↓	78.7	78.1	0.720†
ResNet-38 [89]	wrn38 d8↓	77.9	78.4	2.110†
PSPNet [36]	rn101 d8↓	n/a	78.4	0.720†
Multi Task [90]	rn101 d8↓	n/a	78.5	0.720
TKCN [91]	rn101 d8↓	n/a	79.5	0.720†
DFN [92]	rn101 d32↓	n/a	79.3	0.450†
Mapillary [39]	wrn38 d8↓	78.3	n/a	2.110†
DeepLab v3 [38]	rn101 d8↓	79.3	n/a	0.720†
DeepLabv3+ [80]	x-65 d8↓	79.1	n/a	0.710
DRN [93]	wrn38 d8↓	79.7	79.9	2.110†
DenseASPP [40]	dn161 d8↓	78.9	**80.6**	0.500†
LDN121 64→4	dn121 64↓	**80.1**	**80.0**	**0.066**
LDN161 64→4	dn161 64↓	**80.5**	**80.6**	0.139

Table 4.4: Comparison with the state-of-the-art in efficient inference after training on Cityscapes fine. We report normalized [77] time of FP32 single-scale inference on Titan Xp GPU for 1024×2048 images.

Method	Backbone	$\overline{\text{IoU}}$ Val	$\overline{\text{IoU}}$ Test	Tflop 1Mpx	ms Titan Xp
ERFNet [94]	erfnet d8↓	n/a	69.7	**0.055**	89.0
SwiftNet [77]	rn18 d32↓	n/a	75.5	**0.052**	**22.7**
BiSeNet [95]	rn18 d8↓	n/a	74.7	**0.049**	29.4
LRN18 32→4	rn18 32↓	76.1	n/a	**0.056**	**23.5**
LRN50 32→4	rn50 32↓	77.5	n/a	0.109	52.0
LDN121 dws	dn121 64↓	78.9	n/a	**0.054**	42.6
LDN121 64→4	dn121 64↓	**79.0**	**79.3**	0.066	40.7

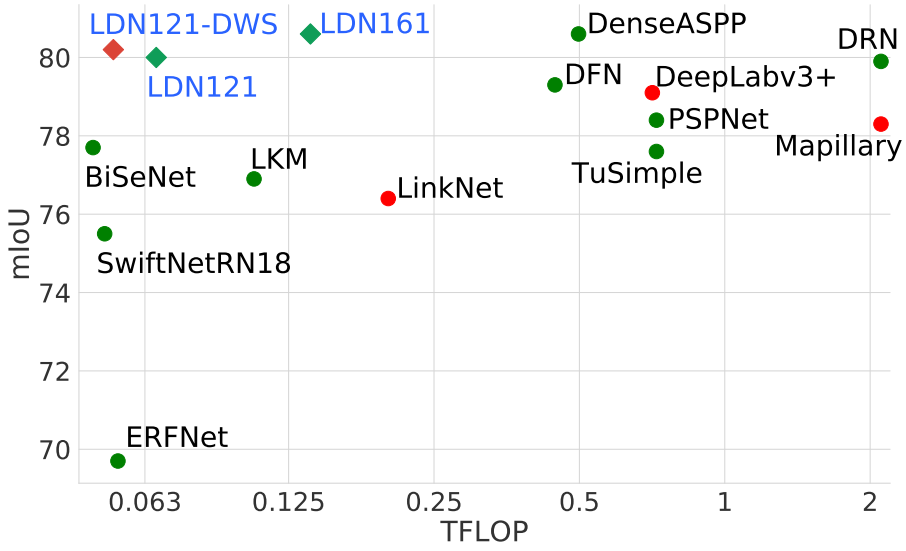

Figure 4.6: Accuracy vs forward pass complexity on Cityscapes test (green) and val (red) for approaches from Table 4.3. LDN121 is the first method to achieve 80% $\overline{\text{IoU}}$ while being applicable in real-time.

Finally, we present some qualitative results on Cityscapes. Figure 4.7 shows predictions from different stages of the upsampling path. Predictions from early layers miss most small objects, however ladder-style upsampling succeeds to recover them after blending with high-resolution features.

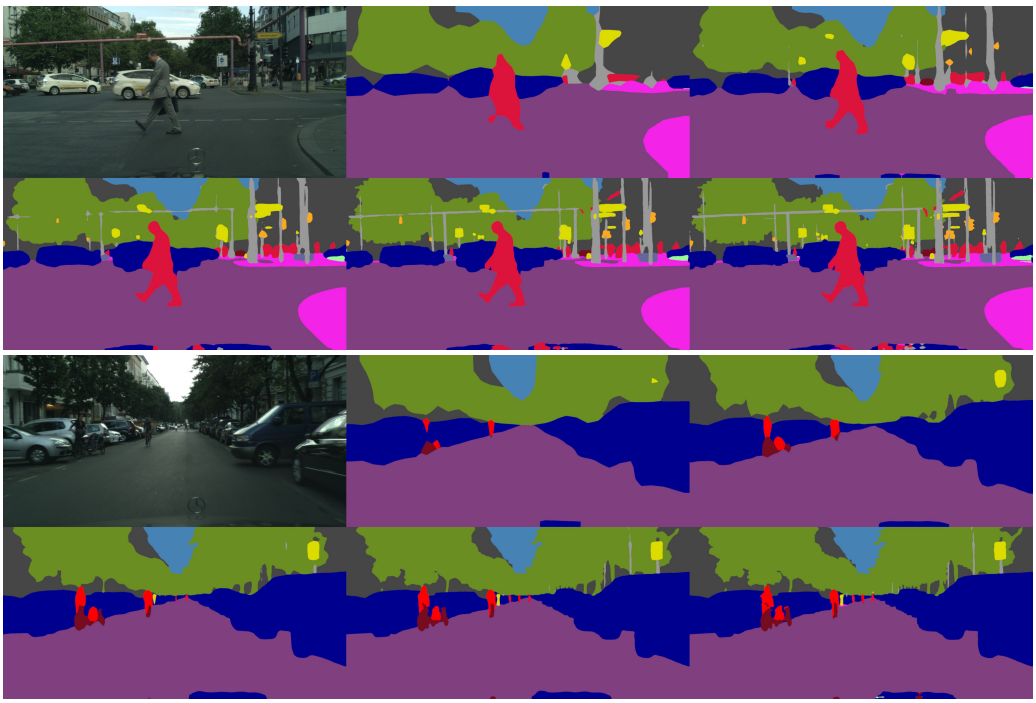

Figure 4.7: Impact of ladder-style upsampling to the accuracy of semantic borders and small objects. We shows predictions recovered from 64×, 32×, 16×, 8×, and 4× subsampled features along the upsampling path.

Figure 4.8 shows the images from the Cityscapes validation set where our model made the most significant errors. Most of the mistakes we observed belong to one of the following types:

- **wall/building/fence**: easy to confuse when covered with posters (1st row); far-away wall can sometimes look like a fence (2nd row); high walls look like buildings (3rd row);
- **road/sidewalk**: road and sidewalk can be hard to distinguish, sometimes causing large labeling errors (4th row); if the lanes are separated with safety traffic signs the lane on the other side is harder to recognize as road (5th row); if the person is crossing the road the part of the road below him has a higher chance of being misclassified as the sidewalk (6th row); if the separation between the road and sidewalk is not visible parts of the sidewalk can get misclassified as road (7th row);
- **vegetation/terrain**: it is hard to draw a clear semantic separation between vegetation (trees and other larger plants) and terrain (grass and other smaller plants); most of the errors of this type are due to inconsistent labeling (8th row);
- **bus/truck/train**: large transport vehicles can sometimes look similar to each other, especially from behind (9th row);
- **motorcycle/bicycle**: can look very similar at a distance with occlusions further increasing the difficulty of distinguishing between them (10th row);
- **person/rider**: occlusions make it harder for the model to separate a person (pedestrians) from a rider (11th row);
- **out of distribution**: Figure 4.9 shows some examples where our model made bad predictions on objects belonging to the classes which were ignored or unseen in the training set.

Figure 4.10 shows images from Cityscapes test in which our best model makes mistakes. It is our impression that most of these errors are due to insufficient context, despite our efforts to enlarge the receptive field. Overall, we achieve the worst IoU on fences (60%) and walls (61%).

Figure 4.11 shows some images from Cityscapes test where our best model performs well in spite of occlusions and large objects. We note that small objects are very well recognized which confirms the merit of ladder-style upsampling. A video demonstration of a single-scale LDN-121 64→4 model is available here: `https://youtu.be/QrB7Np_8GXY`.

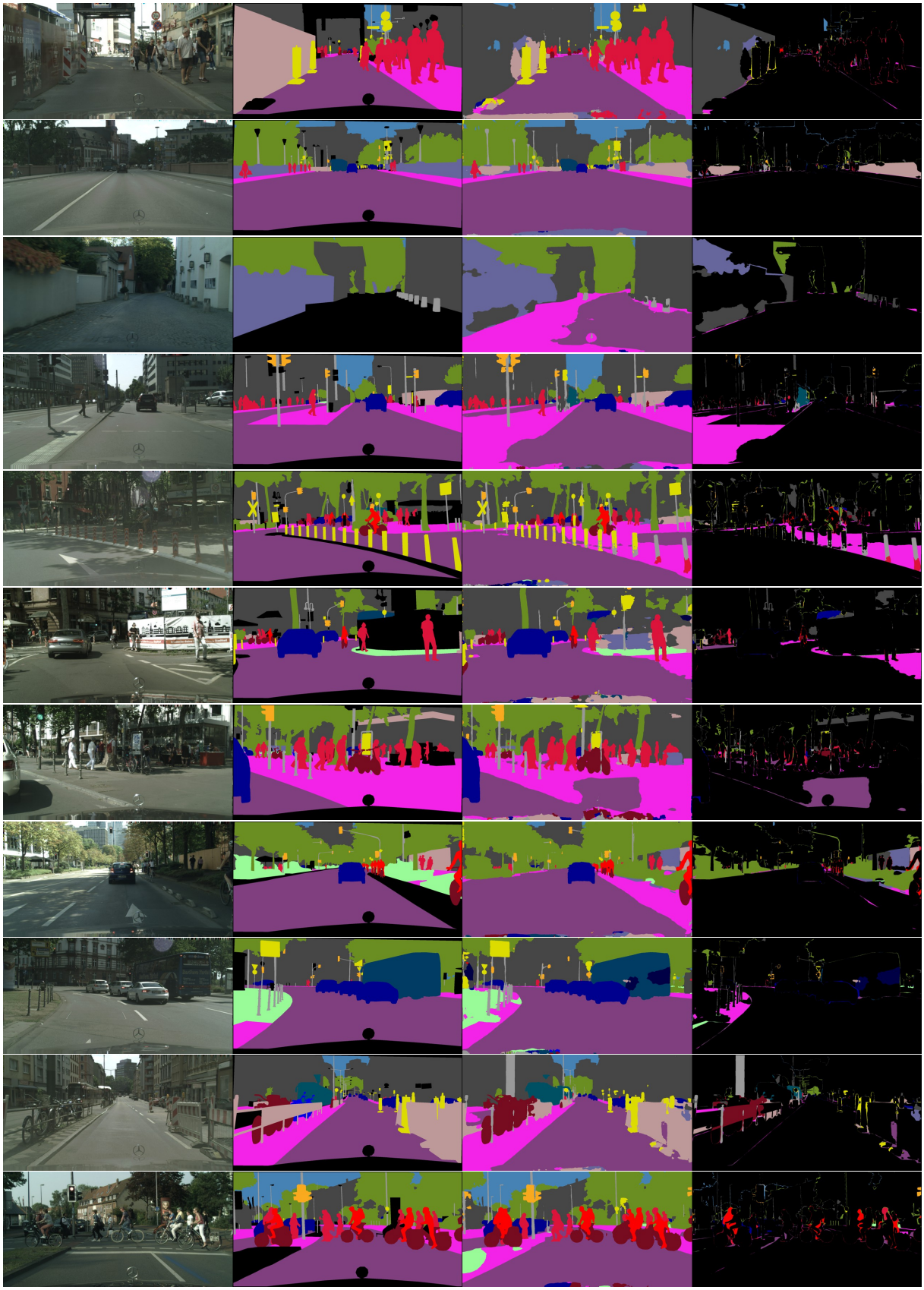

Figure 4.8: Images from Cityscapes validation set where our model made the largest number of errors. Second column shows the groundtruth labels, third column shows the model prediction and the last column shows only the pixels which were misclassified.

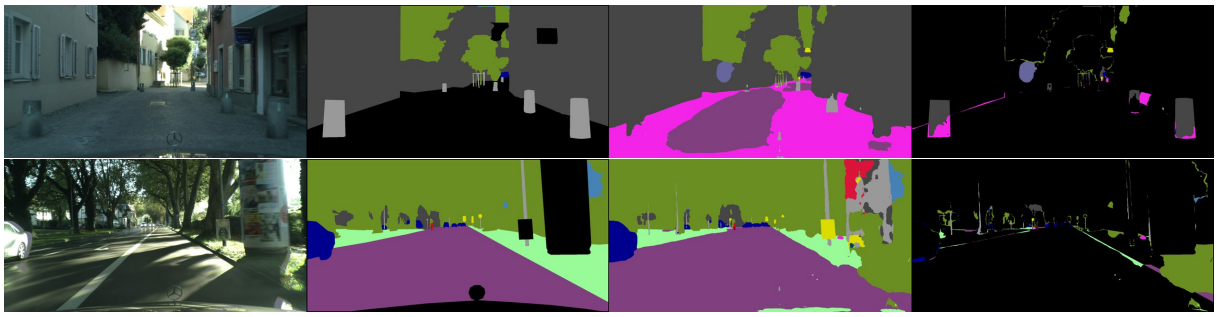

Figure 4.9: Examples of images from Cityscapes validation set where our model encountered out of distributions examples. For instance a type of a parking barrier and advertising posters not seen in the train set.

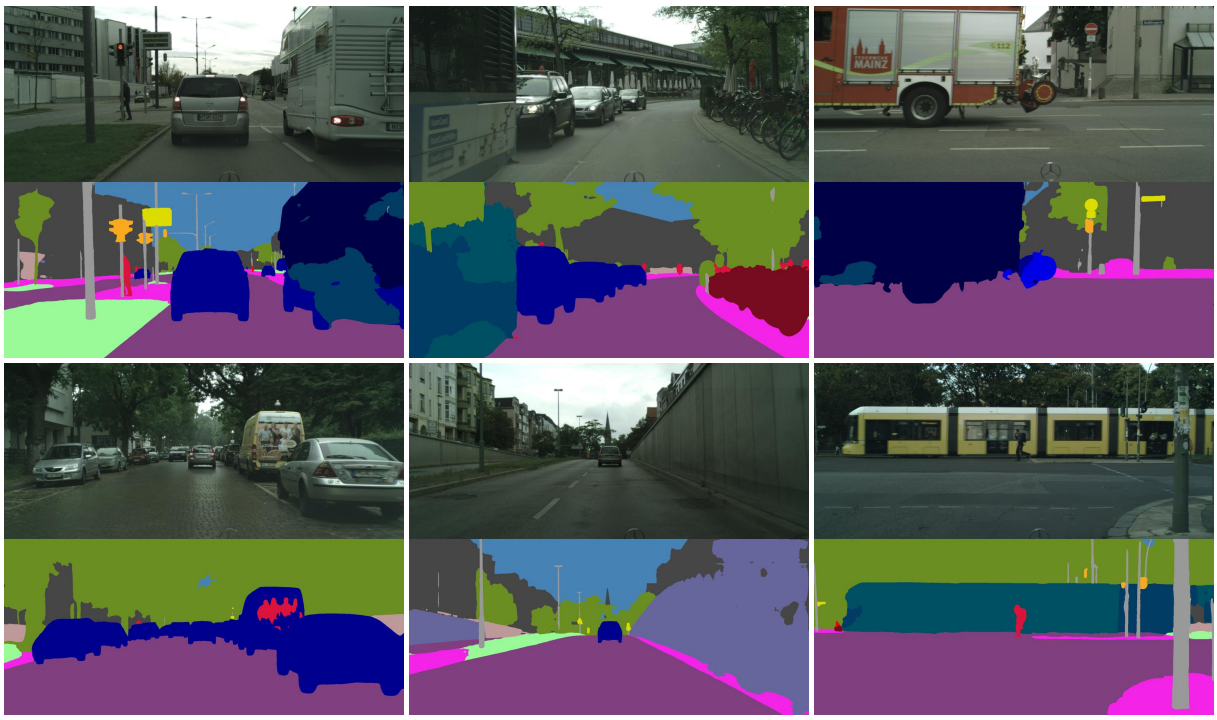

Figure 4.10: Images from Cityscapes test where our model misclassified some parts of the scene. These predictions are produced by the LDN161 64→4 model which achieves 80.6% $\overline{\text{IoU}}$ on the test set.

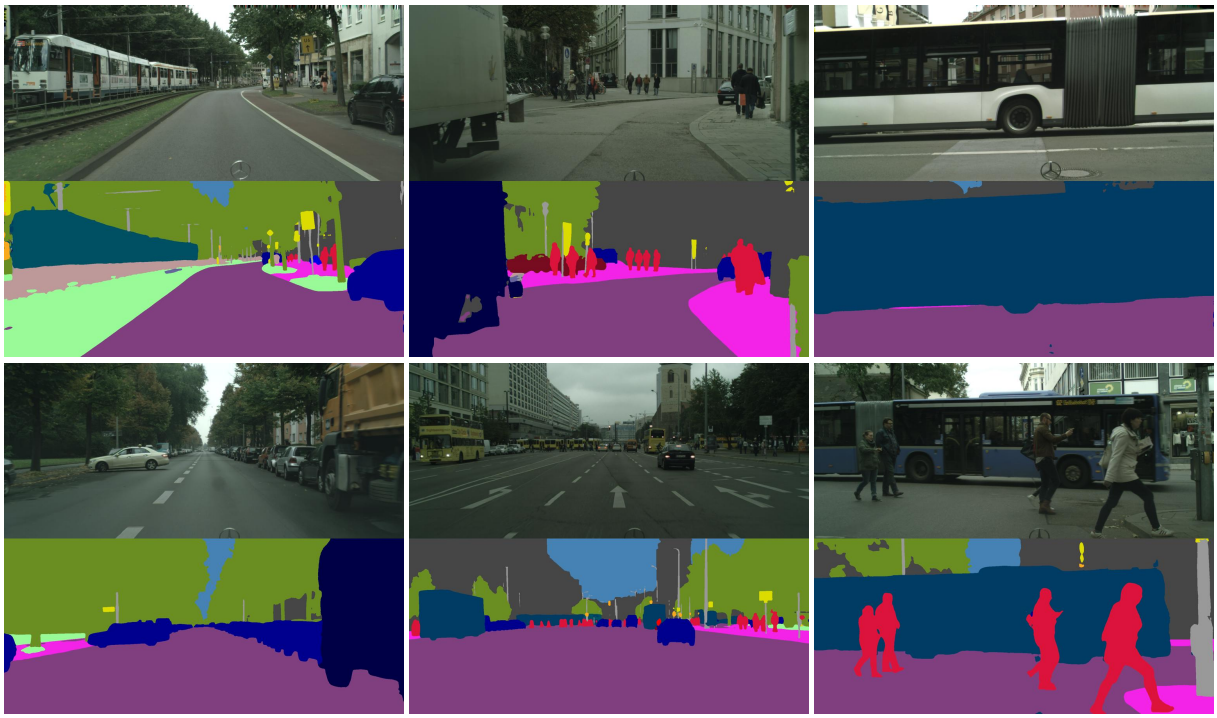

Figure 4.11: Images from Cityscapes test where our best model (LDN161 64→4, 80.6% $\overline{\text{IoU}}$ test) makes no significant errors in spite of large objects, occlusions, small objects and difficult classes.

4.6.3 CamVid

The CamVid dataset contains images of urban road driving scenes. We use the 11-class split from [79], which consists of 367 training, 101 validation, and 233 testing images. The resolution of all images is 720×960. Following the common practice, we incorporate the val subset into train because it is too small and too easy to be useful for validation. We train our models from random initialization (RI), and by fine-tuning the parameters pre-trained on ImageNet (PT). We train on 512×512 crops for 400 epochs with pre-training, and 800 epochs with random init. All other hyperparameters are the same as in Cityscapes experiments. Table 4.5 shows our results on full-resolution CamVid test. The conclusions are similar as on half-resolution Cityscapes val (cf. Table 4.1), which does not surprise us due to similar input resolutions. LDN121 32→4 wins both in the pre-trained and in the random init case, with LDN121 64→4 being the runner-up. Table 4.6 compares our best results with the related work on CamVid test where, to the best of our knowledge, we obtain state-of-the-art results.

Table 4.5: Single-scale inference on full-resolution CamVid test with ImageNet pre-training (PT) and random initialization (RI).

Method	PT $\overline{\text{IoU}}$	RI $\overline{\text{IoU}}$	Model size	FLOP 1MPx
LDN121 32→4	**77.3**	**70.9**	9.0M	75.4G
LDN121 64→4	76.9	68.7	9.5M	66.5G
ResNet18 32→4	73.2	70.0	13.3M	55.7G
ResNet50 32→4	76.1	69.9	26.9M	109.0G
ResNet101 32→4	76.7	69.4	45.9M	186.7G

Table 4.6: Comparison of our models with the state-of-the-art on CamVid test. We use multi-scale inference in experiments on full resolution.

Method	Backbone	ImgNet	Resolution	$\overline{\text{IoU}}$
Tiramisu [72]	DenseNet		half	66.9
FC-DRN [96]	DenseResNet		half	69.4
G-FRNet [97]	VGG-16	✓	half	68.8
BiSeNet [95]	Xception39	✓	full	65.6
ICNet [98]	ResNet-50	✓	full	67.1
BiSeNet [95]	ResNet-18	✓	full	68.7
LDN121 16→2	DenseNet		half	**69.5**
LDN121 32→4	DenseNet		full	**71.9**
LDN121 16→2	DenseNet	✓	half	**75.8**
LDN121 32→4	DenseNet	✓	full	**78.1**

4.6.4 Cross-dataset generalization

We explore the capability of our models to generalize across related datasets. Mapillary Vistas [99] is a large road driving dataset featuring five continents and diverse lighting, seasonal and weather conditions. It contains 18000 training, 2000 validation, and 5000 test images. In our experiments, we remap annotations to 19 Cityscapes classes, and resize all images to width 2048. The KITTI dataset contains road driving images recorded in Karlsruhe [100]. It features the Cityscapes labeling convention and depth reconstruction groundtruth. There are 200 training and 200 test images. All images are 370×1226.

Table 4.7 shows that training only on Cityscapes results in poor generalization due to urban bias and constant acquisition setup. On the other hand, models trained on Vistas generalize much better due to better diversity of the training dataset. Training on both datasets achieves the best results.

Table 4.7: Cross-dataset evaluation on half-resolution images. We train separate models on Cityscapes, Vistas and their union, and show results on validation sets of three driving datasets. These experiments present an older variant of LDN121 64→4 which splits DB4 instead of DB3.

Method	Training dataset	Cityscapes $\overline{\text{IoU}}$	Vistas $\overline{\text{IoU}}$	KITTI $\overline{\text{IoU}}$
LDN121 64→4 s4	Cityscapes	76.0	44.0	59.5
LDN121 64→4 s4	Vistas	68.7	73.0	64.7
LDN121 64→4 s4	Cit. + Vis.	**76.2**	**73.9**	**68.7**

4.6.5 Pascal VOC 2012

PASCAL VOC 2012 [50] contains photographs from private collections. There are 6 indoor classes, 7 vehicles, 7 living beings, and one background class. The dataset contains 1464 train, 1449 validation and 1456 test images of variable size. Following related work, we also train on 10582 images from the AUG set [101]. Due to annotation errors in the AUG labels, we first train for 100 epochs on AUG, and then fine-tune for another 100 epochs on train (or train+val). We use 512×512 crops and divide the learning rate of pretrained weights by 8. All other hyperparameters are the same as in Cityscapes experiments. Table 4.8 shows that our models set the new state-of-the-art among models which do not pre-train on COCO. Examples of predictions on some easy and hard examples are shown in Figure 4.12. We can observe that the Pascal dataset has even larger object scale and pose variance than Cityscapes. A large number of errors are due to the insufficient information when the object scale is large and only the cropped part is visible. For instance sofa and bird images in the bottom row appear on a large scale and are therefore cropped with only a part of the object visible to the model. Furthermore, large pose variance and unseen

context around the object are problematic hard cases. For instance the bicycle on the shelf in the bottom row.

Figure 4.12: Example predictions on the Pascal VOC 2012 test set. Top row shows instances of easy examples for dog, horse, sheep and airplane classes. Bottom row shows instances of hard examples for sofa, bird, cat and bicycle classes. Note that most errors are caused by insufficient information on cropped objects, large object scale and pose variance and rare context around the object.

Table 4.8: Experimental evaluation on Pascal VOC 2012 validation and test.

Method	AUG	MS	Val $\overline{\text{IoU}}$	Test $\overline{\text{IoU}}$
DeepLabv3+ Res101	✓	✓	80.6	n/a
DeepLabv3+ Xcept	✓	✓	81.6	n/a
DDSC [102]	✓		n/a	81.2
AAF [103]	✓	✓	n/a	82.2
PSPNet [36]	✓	✓	n/a	82.6
DFN [92]	✓	✓	80.6	82.7
EncNet [104]	✓	✓	n/a	82.9
LDN121 32→4			76.4	n/a
LDN169 32→4	✓	✓	80.5	81.6
LDN161 32→4			78.6	n/a
LDN161 32→4	✓		80.4	n/a
LDN161 32→4	✓	✓	**81.9**	**83.6**

4.6.6 Ablation experiments on Cityscapes

We interpret the operation of our models by disabling an increasingly large portion of the backbone tail, and evaluating the accuracy of the pruned model. We disable convolutional units by setting their output to zero. In order to allow pruning in all processing blocks, we preserve all transition-down units and 1×1 convolutions in residual connections. The left graph in Figure 4.13 compares LDN121 and LRN50 models after pruning an equal portion of convolutional units from the last two processing blocks of the backbone. We observe that DenseNet converges faster than ResNet, although ladder-style upsampling encourages both backbones to recognize small classes (eg. traffic sign) in early layers. The right graph shows per-class performance of pruned LDN121 models. We observe

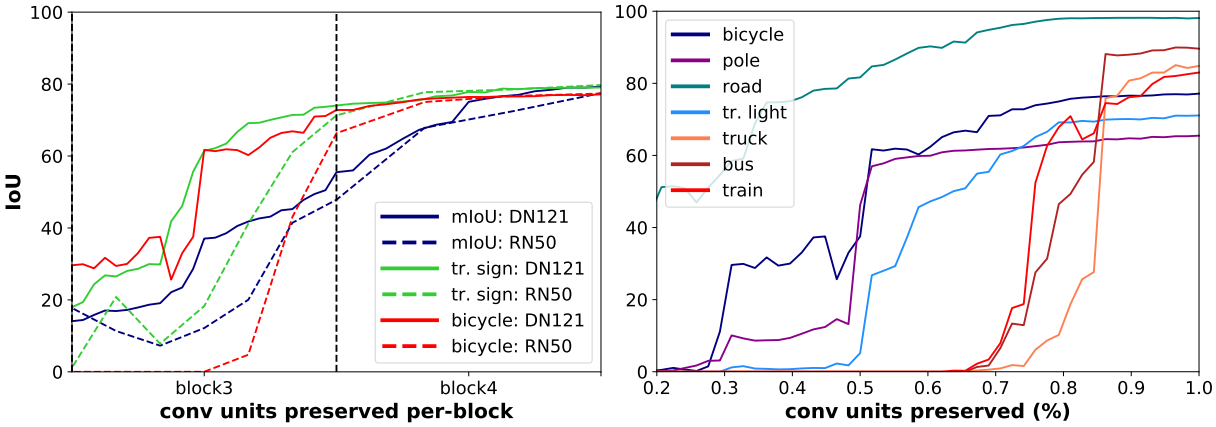

Figure 4.13: Influence of pruning convolutional units from the backbone tail to the Cityscapes val accuracy (there was no re-training). DenseNet is more resilient than ResNet (left). DenseNet recognizes small object classes in early layers, which saves later capacity for large classes (right).

that small and less complex classes (bicycle, pole, traffic sign and road) are recognized by early layers, while hard and large classes (truck, bus and train) are recognized in deeper layers. This can be viewed as a kind of self-regularization: although the back-end of the backbone can have a huge capacity, this capacity is not used for detecting easy classes.

Table 4.9 evaluates the impact of auxiliary loss, SPP, and depthwise separable convolutions on generalization accuracy. The experiment labeled NoSPP replaces the SPP module with a single 3×3 convolution. The resulting 1.5pp performance drop suggests that SPP brings improvement even with 64 times subsampled features. The subsequent experiment shows that the SPP module proposed in [36] does not work well with our training on Cityscapes. We believe that the 1.4pp performance drop is due to inadequate pooling grids and larger feature dimensionality which encourages overfitting. The NoAux model applies the loss only to final predictions. The resulting 1.2pp performance hit suggests that auxiliary loss reduces overfitting to low-level features within the upsampling path. The DWS model reduces the computational complexity by replacing all 3×3 con-

volutions in the upsampling path with depthwise separable convolutions. This improves upsampling efficiency while only marginally decreasing accuracy.

Table 4.9: Impact of auxiliary loss, SPP, and depthwise separable convolutions on generalization accuracy on full-resolution Cityscapes val.

Method	$\overline{\text{IoU}}$	Model size	FLOP 1MPx
LDN121 64→4 NoSPP	77.5	10.2M	66.7G
LDN121 64→4 SPP [36]	77.6	10.6M	66.9G
LDN121 64→4 NoAux	77.8	9.5M	66.5G
LDN121 64→4 DWS	78.6	8.7M	54.2G
LDN121 64→4	**79.0**	9.5M	66.5G

Table 4.10 shows ablation experiments which evaluate the impact of data augmentations on generalization. We observe that random image flip, crop, and scale jitter improve $\overline{\text{IoU}}$ by almost 5pp, and conclude that data augmentation is of great importance for semantic segmentation.

Table 4.10: Impact of data augmentation to the segmentation accuracy ($\overline{\text{IoU}}$) on Cityscapes val while training LDN121 64→4 on full images.

augmentation:	none	flip	flip/crop	flip/crop/scale
accuracy ($\overline{\text{IoU}}$):	74.0	75.7	76.7	79.0

4.6.7 Gradient checkpointing

Table 4.11 explores effects of checkpointing to the memory footprint and the execution speed while training the default LDN model from Figure 4.1. The columns show:

1. the maximum memory allocation while training with batch size 6
2. the maximum batch size we could fit into GPU memory
3. the corresponding training speed in frames per second (FPS).

We start from the straightforward baseline and gradually introduce more and more extensive checkpointing. The checkpointing approaches are designated as follows. The label *cat* refers to the concatenation at the input of a DenseNet unit. The labels 1×1 and 3×3 refer to the first and the second BN-ReLu-conv group within a DenseNet unit. The label *stem* denotes the 7×7 convolution at the very beginning of DenseNet [19], including the following batchnorm, ReLU and max-pool operations. Labels TD and TU correspond to the transition-down and the transition-up blocks. The label *block* refers to the entire processing block (this approach is applicable to most backbones). Parentheses indicate the checkpoint segment. For example, (cat 1×1 3×3) caches only the concatenation inputs,

while the first batchnorm, the 1×1 convolution and the second batchnorm are re-computed during backprop. On the other hand, (cat 1×1) (3×3) means that each convolution is in a separate segment. Here we cache the concatenation inputs and the input to the second batchnorm, while the two batchnorms are recomputed. Consequently, training with (cat 1×1 3×3) accommodates larger batches.

Now we present the most important results. Checkpointing the (cat 1×1) subgraph brings the greatest savings with respect to the baseline (4.5 GB), since it has most feature maps on input. Nevertheless, checkpointing the whole DenseNet unit (cat 1×1 3×3) frees further 3 GB. Finally, checkpointing stem, transition-down and transition-up blocks relieves additional 1.8 GB. Altogether, this results in a more than five-fold reduction of memory requirements, from 11.3 GB to 2.1 GB.

Experiments with the label (block) treat each dense block as a checkpoint segment. This requires more memory than (cat 1×1 3×3) because additional memory needs to be allocated during re-computation. The approach (cat 1×1) (3×3) is similar to the related previous work [19]. These experiments show that the smallest memory footprint is achieved by checkpointing the stem, transition-down and transition-up blocks, as well as each DenseNet unit as a whole.

Table 4.11: Impact of checkpointing to memory footprint and training speed. We train LDN 32→4 on 768×768 images on a Titan Xp with 12 GB RAM.

Checkpointing variant	Memory bs=6 (MB)	Max BS	Train FPS
baseline - no ckpt	11265	6	11.3
(3x3)	10107	6	10.5
(cat 1x1)	6620	10	10.4
(cat 1x1) (3x3)	5552	12	9.7
(block) (stem) (TD) (UP)	3902	16	8.4
(cat 1x1 3x3)	3620	19	10.1
(cat 1x1 3x3) (stem) (TD) (UP)	2106	27	9.2

Table 4.12 shows that our checkpointing approach allows training the LDN161 model with a six-fold increase of batch size with respect to the baseline. On the other hand, the only previous checkpointing technique for semantic segmentation [39] yields only a two-fold increase of batch size.

Table 4.12: Comparison of memory footprint and training speed across various models. We process 768×768 images on a Titan Xp with 12 GB RAM.

Model	Uses Ckpt	Memory bs=6 (MB)	Max BS	Train FPS
LDN161 32→4		20032	3	5.6
ResNet101 32→4		15002	4	7.8
LDN121 32→4		11265	6	11.3
ResNet50 32→4		10070	6	11.6
ResNet18 32→4		3949	17	24.4
LDN161 32→4	✓	3241	19	4.4
LDN121 32→4	✓	2106	27	9.2

4.7 Discussion

We have presented a novel semantic segmentation approach based on DenseNet architecture and ladder-style upsampling. Different than concurrent encoder-decoder approaches, we argue for an asymmetric architecture with thick encoder and thin decoder, which assigns much more capacity to recognition than to localization. In comparison with widely used dilated approaches, our design substantially decreases computational complexity (both space and time) while generalizing better.

The proposed design exhibits outstanding spatial efficiency which however has to be unlocked by recomputing all concats, batchnorms and projections during backprop. This decreases memory footprint for up to 6× while only slightly increasing training time. Thus, LDN121 32→4 can be trained on 768×768 crops with batch size 16 in only 5.3 GB RAM.

We have performed extensive experiments on Cityscapes, CamVid, ROB 2018 and Pascal VOC 2012. We achieve state-of-the-art on Cityscapes test without the coarse training subset (LDN161: 80.6% $\overline{\text{IoU}}$) as well as on Pascal VOC 2012 test without COCO pretraining (LDN161: 83.6% $\overline{\text{IoU}}$). None of the competing approaches is able to train on a single GPU.

To the best of our knowledge, this is the first DenseNet-based approach for efficient dense prediction on 2 MPx images. A TensorRT implementation of our single-scale LDN121 model processes 1024×2048 images at 25 Hz on a single Titan Xp, while achieving 79.3% $\overline{\text{IoU}}$ on Cityscapes test. Suitable directions for future work include further improvements of run-time speed, as well as exploiting the reclaimed memory for dense prediction and forecasting in video.

Chapter 5

Robust Vision Challenge

In this chapter we describe our results while participating in the Robust Vision Challenge (ROB 2018). The goal of the challenge was to measure the robustness of competing models on multiple datasets and on a wide range of vision problems. We competed in the semantic segmentation problem and achieved the 2nd place. The datasets were chosen to cover different domains like indoor vs outdoor scenes. Additionally, one dataset contains out-of-distribution examples where the uncertainty of the model prediction should be taken into account.

Assessing the prediction uncertainty is necessary if we wish to be able to warn downstream processing elements when model predictions are likely to be wrong. Half of the solution consists in detecting image regions which are completely different from the training images and therefore fall in the category of out-of-distribution examples [105]. The other half of the solution is to detect regions which are poorly learned or inherently hard to classify [106], that is to recognize parts of the scene where our models consistently fail to produce correct results.

Furthermore, there is little previous research on semantic segmentation models which are suitable for recognizing different kinds of environments in images with no photographer bias. Before performing experiments presented here we did not know whether such models could be trained without one domain knowledge interfering with another. We also did not know how much capacity is required in order to produce state of the art predictions in different scenarios.

The Robust Vision Challenge provides a good testbed to address these questions. Diversity of the included datasets poses challenges to models which may be biased towards a single dataset while not generalizing well on others. Simultaneous training on diverse datasets provides an opportunity to learn representations which produce good and robust results in a multitude of environments.

We present main findings gathered while participating in the ROB 2018 challenge. We

describe the employed model [4], detail the training procedure and present main insights obtained during our experiments.

5.1 Datasets

We train our common model on the following four training subsets: Cityscapes Fine train+val, WildDash, KITTI train and ScanNet train. Due to limited computing resources and limited time we chose to leave other prospective datasets for future work. Thus, we did not train on Berkeley Deep Drive, Vistas and Cityscapes coarse, although we did initialize our training with parameters learned on ImageNet [7]. The rest of this section provides a brief overview of each of the four training datasets.

5.1.1 Cityscapes

The Cityscapes dataset [49] contains images from the driver's perspective acquired in cities from Germany and neighbouring countries. The dataset provides 2MPx images split into train, val and test subsets, where the semantic labels for the test subset are not publicly available. There are 19 label classes used for evaluation which we train upon. Train and val subsets consist of 2975 and 500 finely annotated images, respectively. The dataset also provides 20 000 coarsely annotated images which we do not use in any of our experiments.

5.1.2 WildDash

The WildDash dataset contains a small selection of worldwide driving images with a strong potential to present difficulties for recognition algorithms. The dataset contains 70 validation and 156 testing images which are grouped into ten specific hazardous scenarios such as blur, windscreen interference, lens distortion etc. The image resolution is 1920x1080px while the semantic annotations follow the Cityscapes labeling policy. As in other datasets, the test labels are not publicly available.

This dataset is unique since the test subset contains a number of heavily distorted and out-of-distribution images whose correct pixel-level predictions may either be the exact class or the class "Void" (both cases are counted as true positives). The negative images must be treated in the same way as the rest of the dataset, which suggests that aspiring models should include a method for detecting out-of-distribution patches in input images.

Due to some labeling inconsistencies (e.g. terrain vs vegetation and car vs truck) we follow the ROB practice and evaluate WildDash performance with the category iIoU metric.

5.1.3 KITTI

The KITTI dataset [48] has been collected in Karlsruhe, Germany while driving through the city itself and the surrounding area. It provides 200 images for training and 200 images for testing at 1242x370px. The dataset uses the Cityscapes labeling policy, same as the previous three driving datasets.

5.1.4 ScanNet

The ScanNet dataset [107] is the only indoor dataset used. It is by far the largest dataset of the four, consisting of nearly 25 000 training and 962 test images. This introduces a large distribution disbalance between indoor and driving labels which needs to be suitably handled. There are 20 semantic classes common to indoor scenery. The image resolution varies, while most images have 1296×968px.

5.2 Method

We use a custom fully convolutional model based on DenseNet-169 [20]. The model features a ladder-style upsampling path [4, 42, 76, 108] which blends high quality semantics of the deep layers with fine spatial detail of the early layers. The model produces logits at 4× subsampled resolution which we upsample to the input resolution with bilinear interpolation, and feed to the usual cross-entropy loss with respect to one-hot groundtruth labels.

The main differences with respect to our previous work [4] are as follows. First, we have replaced all concatenations in the upsampling path with summations. This increased the efficiency of our upsampling path without losing any IoU accuracy in validation experiments. Following that, we have increased the number of convolution filters in the upsampling path from 128 to 256 because we assumed that 128 feature maps could lead to underfitting while training on all four ROB 2018 datasets. Second, we replace the context layer at the end of the downsampling path with a spatial pyramid pooling block very similar to [36]. Third, we remove the auxiliary loss at the end of the downsampling path and replace it with a novel auxiliary loss which we call the pyramid loss. The components of the new loss are defined in terms of softmax predictions obtained from representations obtained right after feature blending units within the upsampling path at 64×, 32×, 16× and 8× subsampled resolution. We do not upsample these auxiliary predictions to the input resolution as in the main loss. Instead, we define the pyramid loss components as cross-entropy between softmax predictions and the groundtruth distribution over class labels in the N×N boxes where N denotes the corresponding subsampling factor.

During training we oversample Cityscapes, KITTI and WildDash images multiple times in order to achieve 2:1 example ratio with respect to ScanNet in each epoch. Mixing outdoor and indoor images into each batch was very important in order to get batchnorm moving population statistics that correctly approximates batch statistics on both tasks. For data augmentation, we apply random scale resize between 0.5 and 2, random crop with 768x768 window size and random horizontal flip with 0.5 probability. These hyper-parameter values are shared for all datasets. We used the Adam optimizer [15] with the base learning rate of $4e^{-4}$ and additionally divide the learning rate by a factor of 4 for the ImageNet pre-trained subset of parameters. The contribution weight of the pyramid auxiliary loss was set to 0.4. We set the batch size to 8 and train the common model for 200k iterations. The training took around 3 days on one Titan Xp GPU.

5.3 Results

We apply the common model to the test subsets of all four datasets, collect the model predictions and map them to the required formats of the individual benchmarks where necessary (Cityscapes). We analyze the obtained results and present the most interesting findings.

5.3.1 Mapping predictions to the dataset formats

A common model for the ROB semantic segmentation challenge has to predict at least 39 object classes: the 19 driving classes from Cityscapes and 20 indoor classes from ScanNet. The benchmark scripts for WildDash, KITI and ScanNet datasets automatically map foreign class indices to the negative classes "Void" (Cityscapes) or "Ignore" (ScanNet). The Cityscapes benchmark is oblivious of ScanNet indices and therefore we had to manually remap ScanNet predictions to the class "Void" (we had very few such pixels as shown in Table 5.1).

Note that the negative classes ("Void" and "Ignore") are separate form the 39 object classes. Predictions of negative classes do not contribute to true positives on Cityscapes, KITTI and ScanNet, however they still may improve performance since they do not count as false positives. However, negative predictions constitute true positives in several WildDash images.

5.3.2 False-positive detections of foreign classes

This group of experiments explores incidence of false negative detections due to predictions of foreign classes. This can be easily evaluated on the test datasets because there is no

overlap between indoor and driving classes. We look at the number of "driving" pixels in the ScanNet test dataset as well as at the number of "indoor" pixels in Cityscapes test, WildDash test and KITTI test. The results are summarized in Table 5.1. The results show that, perhaps surprisingly, cross-dataset training resulted in negligible increase of false positive detections due to sharing the model across different kinds of scenery.

	driving classes (%)	indoor classes (%)
Cityscapes	99.857	0.143
WildDash	97.649	2.351
KITTI	100	0
ScanNet	≈ 0	≈ 100

Table 5.1: Incidence of foreign pixels in the test subsets of the four datasets. The rows correspond to the four datasets while the columns correspond to the two groups of classes. We see that cross-dataset training causes very few false positive pixels and therefore results in a negligible performance hit.

Most foreign pixels in Cityscapes test images are located on the car hood which is ignored during training. Figure 5.1 shows the only Cityscapes test image with a relatively large group of predictions to foreign classes. There were zero detections of foreign classes on KITTI, and only 8 detections of foreign classes on ScanNet. Most of foreign pixels on WildDash test are located in negative images and are therefore treated as true positives (we explore this in more detail later).

5.3.3 Detecting negative WildDash pixels

Closer inspection of WildDash test images revealed that almost all pixels classified as ScanNet occur in the negative WildDash images. We illustrate three such images in Figure 5.2. The figure shows that Cityscapes detections are often correct (people, building) or almost correct (indoor walls as building, lego pavement as road).

Table 5.2 shows the difference in category iIoU performance between our submissions M_DN and LDN2_ROB to the WildDash benchmark. Both submissions correspond to instances of the model described in Section 5.2 trained with similar optimization settings. The submission M_DN maps all indoor predictions to the class Cityscapes "Wall". The submission LDN2_ROB leaves outdoor predictions as they were, which means that the benchmark script automatically maps them to class "Void". By treating ScanNet predictions as the WildDash negative class, LDN2_ROB submission achieved 9.1 percentage points improvement in category iIoU. This improvement gives hope that we could estimate prediction uncertainty by simply assessing the likelihood of the foreign classes. In

Figure 5.1: The only Cityscapes test image in which a large group of pixels was misclassified into indoor classes. The graffiti on the building (white pixels) were classified as the indoor class "wall".

other words, we could train our future models in supervised or semi-supervised manner on diverse datasets and use the prediction of foreign classes (that is, classes that are not supposed to appear in this particular image) as a flag that the predictions are uncertain.

submission ID	ScanNet classes mapped to	negative iIoU category (%)
M_DN	"Wall"	32.2
LDN2_ROB	"Void"	42.8

Table 5.2: Results of our two submissions to the WildDash benchmark (M_DN and LDN2_ROB). LDN2_ROB improves the performance on negative WildDash test images by mapping ScanNet predictions to out-of-distribution pixels.

5.3.4 Reduction of overfitting in the upsampling path

Early experiments showed very poor accuracy of the Ladder-DenseNet architecture on the WildDash test dataset. Further experiments with a simpler model based on bilinear upsampling resulted in better performance. Consequently, we hypothesized that the

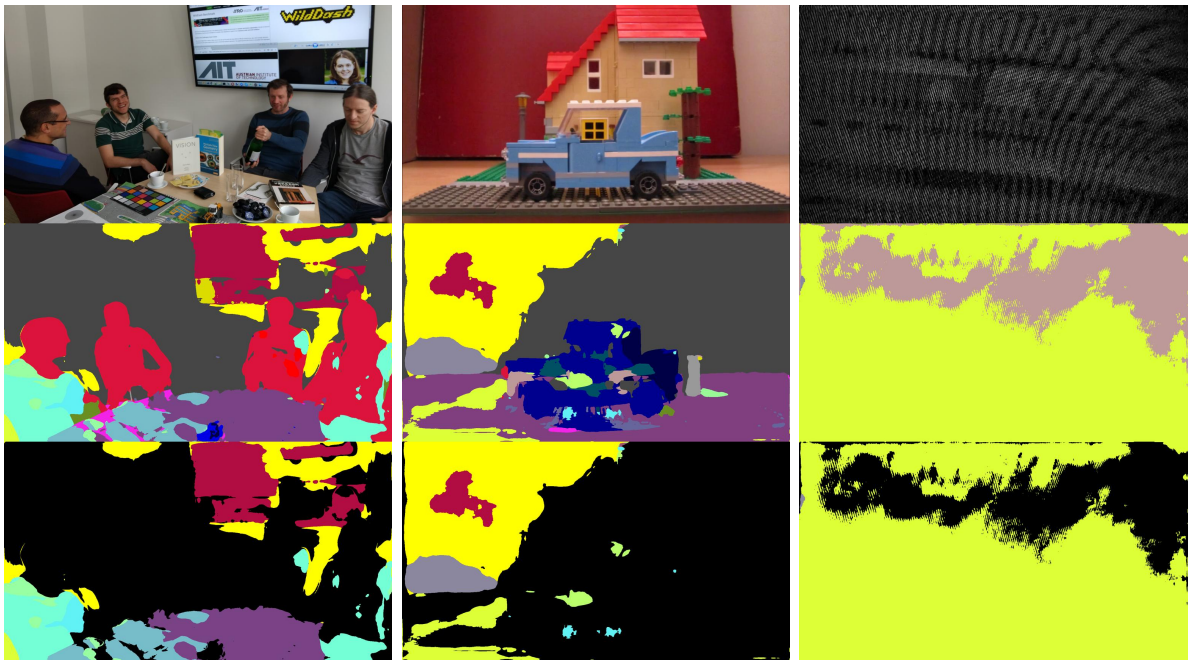

Figure 5.2: Negative WildDash images (top), model predictions (middle) and out-of-distribution pixels for the WildDash dataset (bottom). In the bottom row we colored all pixels classified as Cityscapes classes in black. The remaining colored pixels are treated as the "Void" class during evaluation.

model with the ladder-style upsampling suffers from overfitting in the upsampling path. We attempt to alleviate this problem by regularizing the model with the pyramid loss described in Section 5.2 (i.e. by adding a classification head at each upsampling level), which resulted in significant improvement. We illustrate these effects in Table 5.3 which shows that the recognition accuracy significantly increases when we add pyramid loss. The table also shows that the benefits reproduce on the Berkeley Deep Drive dataset (note that we do not train on Berkeley Deep Drive in any of the experiments).

Further inspection of semantic predictions along the upsampling path showed that some overfitting in the ladder-upsampling remains despite the pyramid loss. We illustrate these effects in Figure 5.3. The image clearly shows that the prediction accuracy gradually decreases as we transition towards finer resolutions (top right). We believe that solving this issue might be an interesting direction for future work.

Finally, we show what happens on WildDash test when we include WildDash val to the training set. The effect is not easy to quantify since WildDash benchmarks allows only three submissions per researcher. We therefore perform qualitative analysis in several WildDash test images and show the results in Figure 5.4. We see that only 70 WildDash val images succeeds to significantly impact the model despite being used along 3500 images from the Cityscapes dataset.

Dataset	WildDash		BDD	
Pyramid loss	No	Yes	No	Yes
Flat	67.2	66.5	70.9	74.6
Construction	18.1	16.8	51.9	52.4
Object	13.4	24.1	29.8	34.3
Nature	72.1	71.9	65.5	65.6
Sky	67.7	66.6	66.9	67.8
Human	30.4	36.0	45.4	46.2
Vehicle	44.1	54.5	75.6	76.3
mIoU	44.7	48.1	58.0	59.6

Table 5.3: Category IoU on WildDash val and Berkeley Deep Drive val for the model trained on Cityscapes only. We observe large improvements on objects, humans and vehicles. Both training and prediction was performed on half image resolution in this experiment.

5.3.5 Overall results

We have submitted the results of our common model to the ROB 2018 challenge under the identifier LDN2_ROB. The obtained results on all four datasets are summarized in Table 5.4.

dataset	metric	our result	best result	our rank
KITTI	class IoU	63.5	69.6	3
ScanNet	class IoU	44.0	48.0	2
Cityscapes	class IoU	77.1	80.2	2
WildDash	category IoU	54.5	59.1	3

Table 5.4: Results of our common model LDN2_ROB at the four semantic segmentation benchmarks.

5.4 Discussion

The presented experiments resulted in several interesting findings. Initial experiments with ladder-style models resulted in very poor cross-dataset performance. Closer inspection revealed that small errors had been multiplying along the upsampling datapath. This likely occured due to blending convolutions being overfit to the Cityscapes urban scenes with ideal weather conditions and a high-quality HDR camera. These effects might be even larger in models with more capacity in the upsampling datapaths. Experiments have

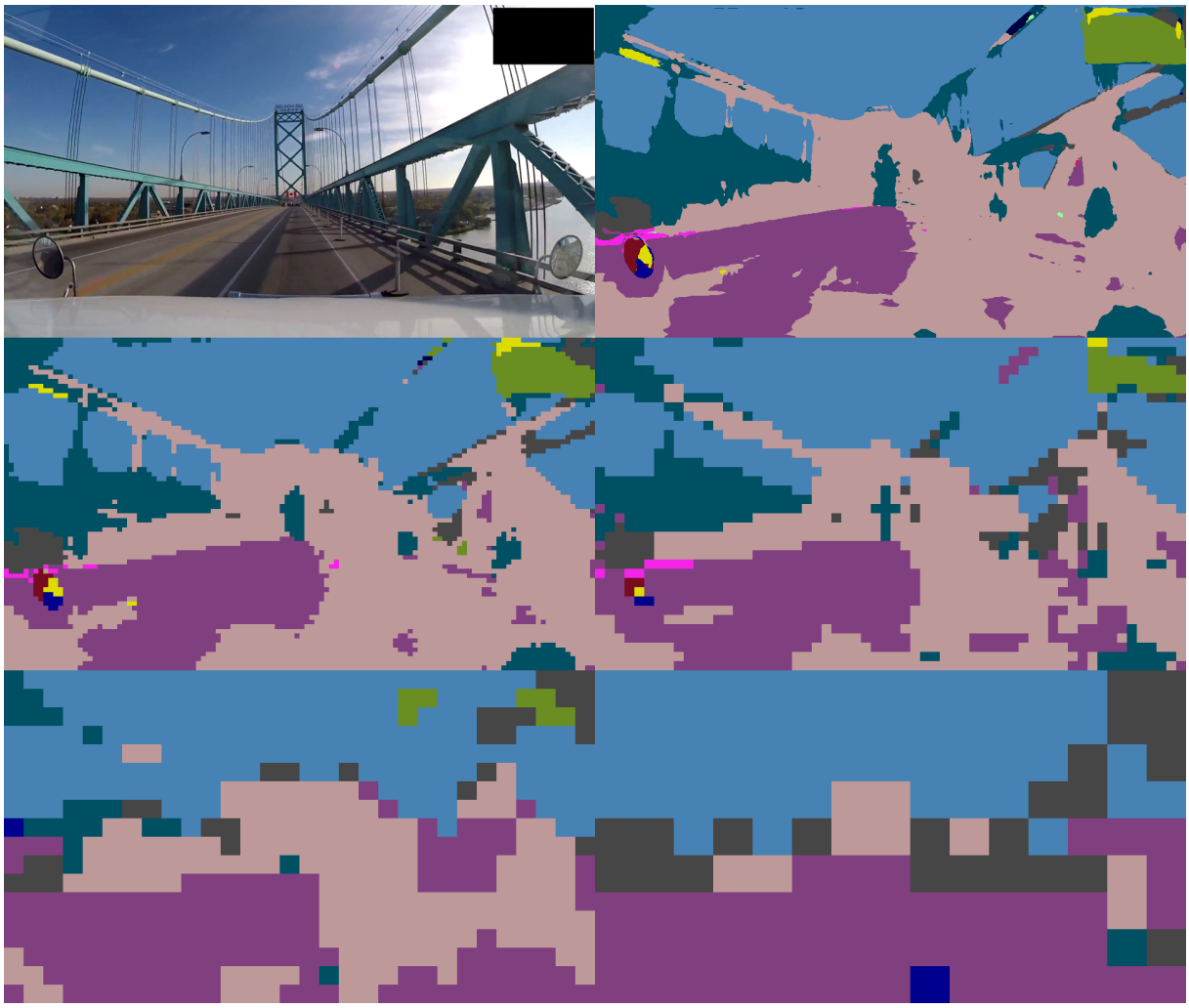

Figure 5.3: Semantic segmentation predictions of a WildDash val image (top left) at different levels of the upsampling path by a model trained on Cityscapes only. The resolution increases from bottom to top and from right to left. We note that the accuracy gets worse as we transition from the coarsest resolution (bottom right) to the finest resolution (top right). Both training and prediction was performed on half image resolution in this experiment.

showed that this problem can be successfully mitigated with suitable auxiliary losses and training on the WildDash val subset.

The second interesting result is that ScanNet training significantly improved recognition of out-of-distribution pixels on WildDash test. In fact, many such pixels were detected as some of the ScanNet classes and were therefore treated as true positive predictions. This raises hopes that future models will be able to detect unusual parts of the scene for free, only by virtue of being trained on a more diverse set of classes.

Further, we found that simultaneous training on multiple datasets resulted in virtually no performance hit with respect to training only on one dataset. In fact, less than 0.01 percent of valid in-distribution pixels in all three driving dataset test sets were recognized as one of the ScanNet indoor classes. Conversely, only 8 out of around billion pixels in ScanNet test were recognized as one of the Cityscapes driving classes.

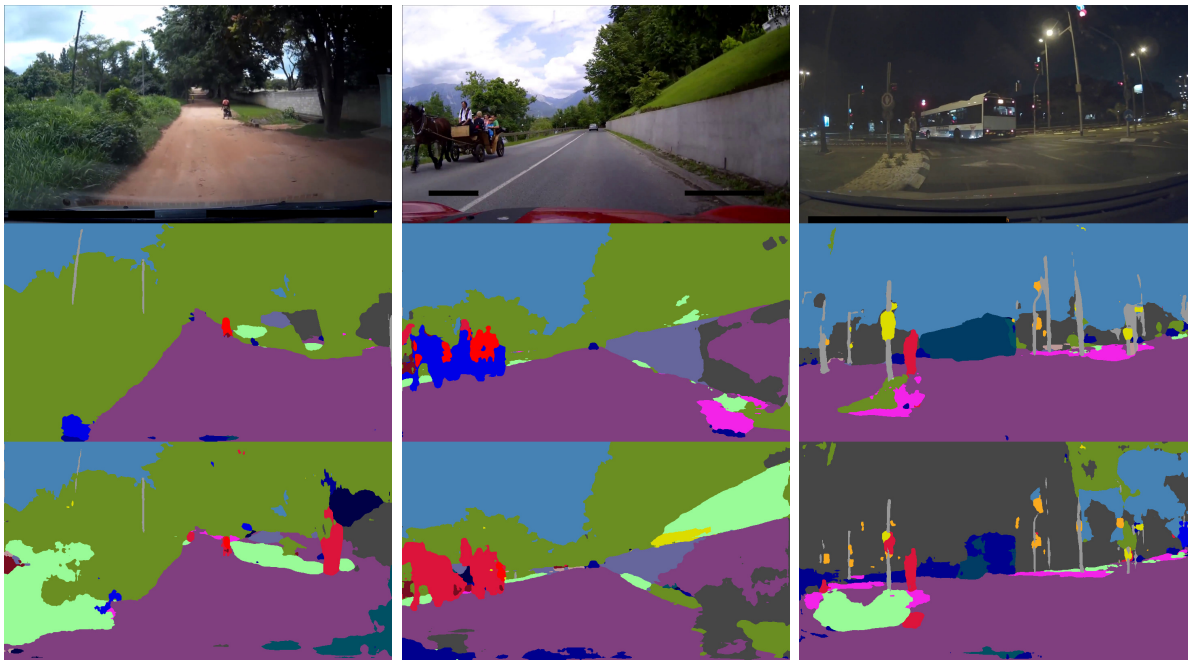

Figure 5.4: Segmentation results when training on Cityscapes + WildDash val (middle row) vs. training on Cityscapes only (bottom row). Both training and predictions was performed on half image resolution in this experiment.

Finally, we found that batch composition represents an important ingredient of cross-dataset training. The training convergence improved substantially when we switched from training on single-dataset batches to training on cross-dataset batches. We hypothesize that the improvement occurred due to more stable training of batchnorm layers.

Chapter 6

Conclusion

In this thesis we have investigated the problem of semantic segmentation of large natural images. The problem of semantic segmentation has many challenges. For instance, one of the problems is that objects appear in the image on a wide range of scales. Models like convolutional networks have to learn to recognize both small and large instances of the same objects. We present a convolutional module for achieving dense scale invariance. Here, the basic idea is to make use of depth information to normalize the scale at which the objects are seen by the ConvNet. The results show the benefits of this approach and are interesting considering how hard it is to improve semantic segmentation with depth data. Furthermore, we contribute a new RGB-D dataset for semantic segmentation of driving scenes. The dataset contains groundtruth semantic segmentations and stereoscopic depth images for 445 annotated hand-picked camera images from the KITTI dataset.

Most current approaches leverage feature extractors from image classification models in order to exploit ImageNet pretraining. However, these feature extractors perform heavy subsampling which makes it quite difficult to produce dense predictions at input resolution. In case of the large input images a very important challenge is how to design computationally and memory efficient model architecture while still preserving quality of predictions comparable to the most accurate models. Another challenge while applying convolutional neural network for semantic segmentation is to ensure that the receptive field of the model is capable of capturing wide context information around a pixel. While most of the previous work focuses only on the accuracy improvements, we also explore a good tradeoff between accuracy and model efficiency.

There are currently two popular approaches to the subsampling problem, either by employing dilated convolutions or by utilizing ladder-style upsampling. In this thesis we argue that ladder-style upsampling is superior to dilated convolution when dealing with large input images. Most current approaches to semantic segmentation are based on residual convolutional units. However, in this thesis we argue that feature extractors

based on densely connected convolutional units present many advantages with respect to their residual counterparts, especially in the case of semantic segmentation of large images. We show that densely connected units require much less parameters and, if implemented properly, require much less memory for training.

Finally, we propose to address semantic segmentation of large images with densely-connected feature extraction and ladder-style upsampling. We show how to train the model on very large images by utilizing gradient checkpointing of a densely connected feature extractor and analyze different checkpointing strategies. We perform extensive experiments on multiple datasets and show that our model is computationally and memory efficient and achieves very good tradeoff between inference speed and overall accuracy. This makes the presented model suitable for real-world applications. Moreover, we describe our participation in the Robust Vision Challenge (RVC 2018). The challenge addressed cross-dataset and cross-domain training of dense prediction models. The datasets were chosen to cover different domains like indoor vs outdoor scenes and out-of-distribution examples. We achieved the 2nd place in the semantic segmentation category. We show that mixed batches ensure stable evolution of batchnorm parameters. Low incidence of foreign predictions suggests that our model succeeded to implicitly learn to distinguish the domains.

Further research can be turned towards making the classification backbone leaner and more efficient for segmentation instead of relying to heavily on architectures fine-tuned for ImageNet problem, as well as utilizing information from videos to improve the accuracy of predictions.

Bibliography

[1] Kreso, I., Causevic, D., Krapac, J., Segvic, S., "Convolutional scale invariance for semantic segmentation", in GCPR, 2016, str. 64–75.

[2] Ros, G., Ramos, S., Granados, M., Bakhtiary, A., Vázquez, D., López, A. M., "Vision-based offline-online perception paradigm for autonomous driving", in 2015 IEEE Winter Conference on Applications of Computer Vision, WACV 2014, Waikoloa, HI, USA, January 5-9, 2015, 2015, str. 231–238.

[3] Kreso, I., Krapac, J., Segvic, S., "Efficient ladder-style densenets for semantic segmentation of large images", IEEE Transactions on Intelligent Transportation Systems, 2020.

[4] Kreso, I., Krapac, J., Segvic, S., "Ladder-style densenets for semantic segmentation of large natural images", in ICCV CVRSUAD, 2017, str. 238–245.

[5] Lecun, Y., Bottou, L., Bengio, Y., Haffner, P., "Gradient-based learning applied to document recognition", in Proceedings of the IEEE, 1998, str. 2278–2324.

[6] Krizhevsky, A., Sutskever, I., Hinton, G. E., "Imagenet classification with deep convolutional neural networks", in NIPS, 2012, str. 1106–1114.

[7] Russakovsky, O., Deng, J., Su, H., Krause, J., Satheesh, S., Ma, S., Huang, Z., Karpathy, A., Khosla, A., Bernstein, M., Berg, A. C., Fei-Fei, L., "ImageNet Large Scale Visual Recognition Challenge", International Journal of Computer Vision (IJCV), Vol. 115, No. 3, 2015, str. 211-252.

[8] Vukotic, V., "Raspoznavanje objekata dubokim neuronskim mrežama", FER, 2014.

[9] LeCun, Y., Bottou, L., Bengio, Y., Haffner, P., "Gradient-based learning applied to document recognition", Proc˙ IEEE, 1998.

[10] Ioffe, S., Szegedy, C., "Batch normalization: Accelerating deep network training by reducing internal covariate shift", in ICML, 2015, str. 448–456.

[11] Santurkar, S., Tsipras, D., Ilyas, A., Madry, A., "How does batch normalization help optimization?", in NIPS, Bengio, S., Wallach, H. M., Larochelle, H., Grauman, K., Cesa-Bianchi, N., Garnett, R., (ur.), 2018, str. 2488–2498.

[12] Wu, Y., He, K., "Group normalization", International Journal of Computer Vision (IJCV), Vol. 128, 2020.

[13] Maas, A. L., Hannun, A. Y., Ng, A. Y., "Rectifier nonlinearities improve neural network acoustic models", in ICML, 2013.

[14] Srivastava, N., Hinton, G. E., Krizhevsky, A., Sutskever, I., Salakhutdinov, R., "Dropout: a simple way to prevent neural networks from overfitting", Journal of Machine Learning Research, Vol. 15, No. 1, 2014, str. 1929–1958.

[15] Kingma, D. P., Ba, J., "Adam: A method for stochastic optimization", CoRR, Vol. abs/1412.6980, 2014.

[16] Simonyan, K., Zisserman, A., "Very deep convolutional networks for large-scale image recognition", in ICLR, 2014, str. 1–16.

[17] Szegedy, C., Liu, W., Jia, Y., Sermanet, P., Reed, S. E., Anguelov, D., Erhan, D., Vanhoucke, V., Rabinovich, A., "Going deeper with convolutions", in CVPR, 2015, str. 1–9.

[18] He, K., Zhang, X., Ren, S., Sun, J., "Deep residual learning for image recognition", in CVPR, 2016, str. 770–778.

[19] Huang, G., Liu, Z., Pleiss, G., Maaten, L. V. D., Weinberger, K., "Convolutional networks with dense connectivity", IEEE Trans. Pattern Anal. Mach. Intell., 2019.

[20] Huang, G., Liu, Z., van der Maaten, L., Weinberger, K. Q., "Densely connected convolutional networks", in CVPR, 2017, str. 2261–2269.

[21] He, K., Zhang, X., Ren, S., Sun, J., "Identity mappings in deep residual networks", in ECCV, 2016, str. 630–645.

[22] Li, H., Xu, Z., Taylor, G., Studer, C., Goldstein, T., "Visualizing the loss landscape of neural nets", in NeurIPS, Bengio, S., Wallach, H. M., Larochelle, H., Grauman, K., Cesa-Bianchi, N., Garnett, R., (ur.), 2018, str. 6391–6401.

[23] Chao, P., Kao, C., Ruan, Y., Huang, C., Lin, Y., "Hardnet: A low memory traffic network", in ICCV, 2019, str. 3551–3560.

[24] Huang, G., Liu, S., van der Maaten, L., Weinberger, K. Q., "Condensenet: An efficient densenet using learned group convolutions", in CVPR, 2018.

[25] Huang, G., Chen, D., Li, T., Wu, F., van der Maaten, L., Weinberger, K. Q., "Multi-scale dense networks for resource efficient image classification", in ICLR, 2018.

[26] Pleiss, G., Chen, D., Huang, G., Li, T., van der Maaten, L., Weinberger, K. Q., "Memory-efficient implementation of densenets", CoRR, Vol. abs/1707.06990, 2017.

[27] Mottaghi, R., Chen, X., Liu, X., Cho, N., Lee, S., Fidler, S., Urtasun, R., Yuille, A. L., "The role of context for object detection and semantic segmentation in the wild", in CVPR 2014, 2014, str. 891–898.

[28] Shotton, J., Winn, J. M., Rother, C., Criminisi, A., "Textonboost for image understanding: Multi-class object recognition and segmentation by jointly modeling texture, layout, and context", International Journal of Computer Vision, Vol. 81, No. 1, 2009, str. 2–23.

[29] Csurka, G., Perronnin, F., "An efficient approach to semantic segmentation", International Journal of Computer Vision, Vol. 95, No. 2, 2011, str. 198–212.

[30] Farabet, C., Couprie, C., Najman, L., LeCun, Y., "Learning hierarchical features for scene labeling", IEEE Trans. Pattern Anal. Mach. Intell., Vol. 35, No. 8, 2013, str. 1915–1929.

[31] Krähenbühl, P., Koltun, V., "Efficient inference in fully connected crfs with gaussian edge potentials", in NIPS, 2011, str. 109–117.

[32] Lin, G., Shen, C., van den Hengel, A., Reid, I. D., "Efficient piecewise training of deep structured models for semantic segmentation", in CVPR, 2016, str. 3194–3203.

[33] Lin, T., Dollár, P., Girshick, R. B., He, K., Hariharan, B., Belongie, S. J., "Feature pyramid networks for object detection", in CVPR, 2017, str. 936–944.

[34] Wang, J., Sun, K., Cheng, T., Jiang, B., Deng, C., Zhao, Y., Liu, D., Mu, Y., Tan, M., Wang, X., Liu, W., Xiao, B., "Deep high-resolution representation learning for visual recognition", CoRR, Vol. abs/1908.07919, 2019.

[35] Yu, F., Koltun, V., "Multi-scale context aggregation by dilated convolutions", in ICLR, 2016, str. 1–9.

[36] Zhao, H., Shi, J., Qi, X., Wang, X., Jia, J., "Pyramid scene parsing network", in ICCV, 2017.

[37] Chen, L., Papandreou, G., Kokkinos, I., Murphy, K., Yuille, A. L., "Deeplab: Semantic image segmentation with deep convolutional nets, atrous convolution, and fully connected crfs", IEEE Trans. Pattern Anal. Mach. Intell., Vol. 40, No. 4, 2018, str. 834–848.

[38] Chen, L., Papandreou, G., Schroff, F., Adam, H., "Rethinking atrous convolution for semantic image segmentation", CoRR, Vol. abs/1706.05587, 2017.

[39] Rota Bulò, S., Porzi, L., Kontschieder, P., "In-place activated batchnorm for memory-optimized training of DNNs", in CVPR, June 2018.

[40] Yang, M., Yu, K., Zhang, C., Li, Z., Yang, K., "DenseASPP for semantic segmentation in street scenes", in CVPR, 2018, str. 3684–3692.

[41] Kreso, I., Krapac, J., Segvic, S., "Efficient ladder-style densenets for semantic segmentation of large images", CoRR, Vol. abs/1905.05661, 2019.

[42] Ronneberger, O., Fischer, P., Brox, T., "U-net: Convolutional networks for biomedical image segmentation", CoRR, Vol. abs/1505.04597, 2015.

[43] Viola, P. A., Jones, M. J., "Robust real-time face detection", International Journal of Computer Vision, Vol. 57, No. 2, 2004, str. 137–154.

[44] Divvala, S. K., Hoiem, D., Hays, J., Efros, A. A., Hebert, M., "An empirical study of context in object detection", in 2009 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR 2009), 20-25 June 2009, Miami, Florida, USA, 2009, str. 1271–1278.

[45] Chen, L., Yang, Y., Wang, J., Xu, W., Yuille, A. L., "Attention to scale: Scale-aware semantic image segmentation", in IEEE Conference on Computer Vision and Pattern Recognition, CVPR, Las Vegas, Nevada, 2016, str. to appear.

[46] Long, J., Shelhamer, E., Darrell, T., "Fully convolutional networks for semantic segmentation", in CVPR, 2015, str. 3431–3440.

[47] Mostajabi, M., Yadollahpour, P., Shakhnarovich, G., "Feedforward semantic segmentation with zoom-out features", in CVPR, 2015, str. 3376–3385.

[48] Geiger, A., Lenz, P., Stiller, C., Urtasun, R., "Vision meets robotics: The kitti dataset", International Journal of Robotics Research (IJRR), 2013.

[49] Cordts, M., Omran, M., Ramos, S., Scharwächter, T., Enzweiler, M., Benenson, R., Franke, U., Roth, S., Schiele, B., "The cityscapes dataset", in CVPRW, 2015.

[50] Everingham, M., Eslami, S. M. A., Gool, L. V., Williams, C. K. I., Winn, J. M., Zisserman, A., "The pascal visual object classes challenge: A retrospective", International Journal of Computer Vision, Vol. 111, No. 1, 2015, str. 98–136.

[51] Ladicky, L., Shi, J., Pollefeys, M., "Pulling things out of perspective", in 2014 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2014, Columbus, OH, USA, June 23-28, 2014, 2014, str. 89–96.

[52] Eigen, D., Fergus, R., "Predicting depth, surface normals and semantic labels with a common multi-scale convolutional architecture", in 2015 IEEE International Conference on Computer Vision, ICCV 2015, Santiago, Chile, December 7-13, 2015, 2015, str. 2650–2658.

[53] Martinovic, A., Knopp, J., Riemenschneider, H., Gool, L. V., "3d all the way: Semantic segmentation of urban scenes from start to end in 3d", in IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2015, Boston, MA, USA, June 7-12, 2015, 2015.

[54] Banica, D., Sminchisescu, C., "Second-order constrained parametric proposals and sequential search-based structured prediction for semantic segmentation in RGB-D images", in IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2015, Boston, MA, USA, June 7-12, 2015, 2015, str. 3517–3526.

[55] Chen, X., Kundu, K., Zhu, Y., Berneshawi, A., Ma, H., Fidler, S., Urtasun, R., "3d object proposals for accurate object class detection", in NIPS, 2015.

[56] Hirschmüller, H., "Stereo vision in structured environments by consistent semi-global matching", in 2006 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR 2006), 17-22 June 2006, New York, NY, USA, 2006, str. 2386–2393.

[57] Zbontar, J., LeCun, Y., "Stereo matching by training a convolutional neural network to compare image patches", CoRR, Vol. abs/1510.05970, 2015.

[58] Chen, L., Papandreou, G., Kokkinos, I., Murphy, K., Yuille, A. L., "Semantic image segmentation with deep convolutional nets and fully connected crfs", in International Conference on Learning Representations, ICLR 2015, San Diego, California., 2014.

[59] Sermanet, P., Eigen, D., Zhang, X., Mathieu, M., Fergus, R., LeCun, Y., "Overfeat: Integrated recognition, localization and detection using convolutional networks", in ICLR, 2014, str. 1–16.

[60] Jia, Y., Shelhamer, E., Donahue, J., Karayev, S., Long, J., Girshick, R. B., Guadarrama, S., Darrell, T., "Caffe: Convolutional architecture for fast feature embedding", in Proceedings of the ACM International Conference on Multimedia, MM '14, Orlando, FL, USA, November 03 - 07, 2014, 2014, str. 675–678.

[61] Noh, H., Hong, S., Han, B., "Learning deconvolution network for semantic segmentation", in ICCV, 2015, str. 1520–1528.

[62] Kendall, A., Badrinarayanan, V., Cipolla, R., "Bayesian segnet: Model uncertainty in deep convolutional encoder-decoder architectures for scene understanding", CoRR, Vol. abs/1511.02680, 2015.

[63] Arnab, A., Jayasumana, S., Zheng, S., Torr, P. H. S., "Higher order potentials in end-to-end trainable conditional random fields", CoRR, Vol. abs/1511.08119, 2015.

[64] Kreso, I., Segvic, S., "KITTI semantic segmentation dataset", https://data.mendeley.com/datasets/3bmmnfb4bp/, 2017.

[65] Kreso, I., Segvic, S., "KITTI semantic segmentation dataset", http://multiclod.zemris.fer.hr/kitti_semseg_unizg.shtml, 2017.

[66] Kingma, D. P., Ba, J., "Adam: A method for stochastic optimization", CoRR, Vol. abs/1412.6980, 2014.

[67] Everingham, M., Gool, L., Williams, C. K., Winn, J., Zisserman, A., "The pascal visual object classes (voc) challenge", Int. J. Comput. Vision, 2010.

[68] Veit, A., Wilber, M. J., Belongie, S. J., "Residual networks behave like ensembles of relatively shallow networks", in NIPS, 2016, str. 550–558.

[69] Chen, Y., Li, J., Xiao, H., Jin, X., Yan, S., Feng, J., "Dual path networks", in NIPS, 2017, str. 4470–4478.

[70] Chen, T., Xu, B., Zhang, C., Guestrin, C., "Training deep nets with sublinear memory cost", CoRR, Vol. abs/1604.06174, 2016.

[71] Ronneberger, O., Fischer, P., Brox, T., "U-net: Convolutional networks for biomedical image segmentation", in MICCAI, 2015, str. 234–241.

[72] Jégou, S., Drozdzal, M., Vázquez, D., Romero, A., Bengio, Y., "The one hundred layers tiramisu: Fully convolutional densenets for semantic segmentation", CoRR, Vol. abs/1611.09326, 2016.

[73] Ghiasi, G., Fowlkes, C. C., "Laplacian pyramid reconstruction and refinement for semantic segmentation", in ECCV, 2016, str. 519–534.

[74] Lin, G., Milan, A., Shen, C., Reid, I. D., "Refinenet: Multi-path refinement networks for high-resolution semantic segmentation", in CVPR, 2017.

[75] Peng, C., Zhang, X., Yu, G., Luo, G., Sun, J., "Large kernel matters - improve semantic segmentation by global convolutional network", in CVPR, July 2017.

[76] Valpola, H., "From neural PCA to deep unsupervised learning", CoRR, Vol. abs/1411.7783, 2014.

[77] Orsic, M., Kreso, I., Bevandic, P., Segvic, S., "In defense of pre-trained imagenet architectures for real-time semantic segmentation of road-driving images", in CVPR, 2019.

[78] Shelhamer, E., Long, J., Darrell, T., "Fully convolutional networks for semantic segmentation", IEEE Trans. Pattern Anal. Mach. Intell., Vol. 39, No. 4, 2017, str. 640–651.

[79] Badrinarayanan, V., Kendall, A., Cipolla, R., "Segnet: A deep convolutional encoder-decoder architecture for image segmentation", IEEE Trans. Pattern Anal. Mach. Intell., Vol. 39, No. 12, 2017, str. 2481–2495.

[80] Chen, L., Zhu, Y., Papandreou, G., Schroff, F., Adam, H., "Encoder-decoder with atrous separable convolution for semantic image segmentation", in ECCV, 2018, str. 833–851.

[81] Pohlen, T., Hermans, A., Mathias, M., Leibe, B., "Full-resolution residual networks for semantic segmentation in street scenes", in CVPR, July 2017.

[82] Islam, M. A., Rochan, M., Neil D. B., B. Wang, Y., "Gated feedback refinement network for dense image labeling", in CVPR, July 2017.

[83] He, K., Gkioxari, G., Dollár, P., Girshick, R. B., "Mask R-CNN", in ICCV, 2017, str. 2980–2988.

[84] Zhao, H., Zhang, Y., Liu, S., Shi, J., Loy, C. C., Lin, D., Jia, J., "Psanet: Point-wise spatial attention network for scene parsing", in ECCV, 2018, str. 270–286.

[85] Lazebnik, S., Schmid, C., Ponce, J., "Beyond bags of features: Spatial pyramid matching for recognizing natural scene categories", in CVPR, 2006, str. 2169–2178.

[86] He, K., Zhang, X., Ren, S., Sun, J., "Spatial pyramid pooling in deep convolutional networks for visual recognition", IEEE Trans. Pattern Anal. Mach. Intell., Vol. 37, No. 9, 2015, str. 1904–1916.

[87] Wang, P., Chen, P., Yuan, Y., Liu, D., Huang, Z., Hou, X., Cottrell, G., "Understanding Convolution for Semantic Segmentation", CoRR, Vol. abs/1702.08502, 2017.

[88] Zhang, R., Tang, S., Zhang, Y., Li, J., Yan, S., "Scale-adaptive convolutions for scene parsing", in ICCV, Oct 2017.

[89] Wu, Z., Shen, C., van den Hengel, A., "Wider or deeper: Revisiting the resnet model for visual recognition", CoRR, Vol. abs/1611.10080, 2016.

[90] Kendall, A., Gal, Y., Cipolla, R., "Multi-task learning using uncertainty to weigh losses for scene geometry and semantics", in CVPR, 2018, str. 7482–7491.

[91] Wu, T., Tang, S., Zhang, R., Cao, J., Li, J., "Tree-structured kronecker convolutional network for semantic segmentation", CoRR, Vol. abs/1812.04945, 2018.

[92] Yu, C., Wang, J., Peng, C., Gao, C., Yu, G., Sang, N., "Learning a discriminative feature network for semantic segmentation", in CVPR, 2018, str. 1857–1866.

[93] Zhuang, Y., Yang, F., Tao, L., Ma, C., Zhang, Z., Li, Y., Jia, H., Xie, X., Gao, W., "Dense relation network: Learning consistent and context-aware representation for semantic image segmentation", in ICIP, 2018, str. 3698-3702.

[94] Romera, E., Alvarez, J. M., Bergasa, L. M., Arroyo, R., "Erfnet: Efficient residual factorized convnet for real-time semantic segmentation", IEEE Trans. Intelligent Transportation Systems, Vol. 19, No. 1, 2018, str. 263–272.

[95] Yu, C., Wang, J., Peng, C., Gao, C., Yu, G., Sang, N., "Bisenet: Bilateral segmentation network for real-time semantic segmentation", in ECCV, 2018, str. 334–349.

[96] Casanova, A., Cucurull, G., Drozdzal, M., Romero, A., Bengio, Y., "On the iterative refinement of densely connected representation levels for semantic segmentation", CoRR, Vol. abs/1804.11332, 2018.

[97] Islam, M. A., Rochan, M., Naha, S., Bruce, N. D. B., Wang, Y., "Gated feedback refinement network for coarse-to-fine dense semantic image labeling", CoRR, Vol. abs/1806.11266, 2018.

[98] Zhao, H., Qi, X., Shen, X., Shi, J., Jia, J., "Icnet for real-time semantic segmentation on high-resolution images", in ECCV, Vol. 11207, 2018, str. 418–434.

[99] Neuhold, G., Ollmann, T., Rota Bulò, S., Kontschieder, P., "The mapillary vistas dataset for semantic understanding of street scenes", in ICCV, 2017.

[100] Alhaija, H., Mustikovela, S., Mescheder, L., Geiger, A., Rother, C., "Augmented reality meets computer vision: Efficient data generation for urban driving scenes", International Journal of Computer Vision (IJCV), 2018.

[101] Hariharan, B., Arbelaez, P., Bourdev, L. D., Maji, S., Malik, J., "Semantic contours from inverse detectors", in ICCV, 2011, str. 991–998.

[102] Bilinski, P., Prisacariu, V., "Dense decoder shortcut connections for single-pass semantic segmentation", in CVPR, 2018, str. 6596–6605.

[103] Ke, T., Hwang, J., Liu, Z., Yu, S. X., "Adaptive affinity fields for semantic segmentation", in ECCV, 2018, str. 605–621.

[104] Zhang, H., Dana, K. J., Shi, J., Zhang, Z., Wang, X., Tyagi, A., Agrawal, A., "Context encoding for semantic segmentation", in CVPR, 2018, str. 7151–7160.

[105] Hendrycks, D., Gimpel, K., "A baseline for detecting misclassified and out-of-distribution examples in neural networks", in ICLR, 2017.

[106] Kendall, A., Gal, Y., "What uncertainties do we need in bayesian deep learning for computer vision?", in NeurIPS, 2017, str. 5574–5584.

[107] Dai, A., Chang, A. X., Savva, M., Halber, M., Funkhouser, T. A., Nießner, M., "Scannet: Richly-annotated 3d reconstructions of indoor scenes", in CVPR, 2017, str. 2432–2443.

[108] Lin, T., Dollár, P., Girshick, R. B., He, K., Hariharan, B., Belongie, S. J., "Feature pyramid networks for object detection", in CVPR, 2017, str. 936–944.

Biography

Ivan Krešo was born in Sisak in 1989. He received his MSc degree in computer science from the University of Zagreb in 2013. After that, he spent six years at the Faculty of Electrical Engineering and Computing, University of Zagreb as a PhD student and teaching assistant where his work time was divided between scientific research and teaching. During that time he participated in creating the deep learning course. He was a teaching assistant in courses related to deep learning, computer architecture, design patterns in programming and computer graphics. Furthermore, he participated in two research projects: Multiclass Object Detection (MULTICLOD) and Computer Vision Innovations for Safe Traffic (VISTA) Currently, he works in the industry as a data scientist at RealNetworks. His research interests include object classification and dense prediction for semantic segmentation and detection, as well as self-supervised and semi-supervised learning.

List of publications

International journal papers

1. Krešo, I., Krapac, J., Šegvić, S., "Efficient Ladder-Style DenseNets for Semantic Segmentation of Large Images", IEEE Transactions on Intelligent Transportation Systems, 2020.

International conference and workshop publications

1. Krešo, I., Ševrović, M., Šegvić, S., "Convolutional scale invariance for semantic segmentation", German Conference on Pattern Recognition (GCPR), 2016.
2. Krešo, I., Krapac, J., Šegvić, S., "Ladder-Style DenseNets for Semantic Segmentation of Large Natural Images", International Conference on Computer Vision Workshop (ICCVW), 2017.
3. Krešo, I., Oršić, M., Bevandić, P., Šegvić, S., "Robust semantic segmentation with ladder-densenet models", Robust Vision Challenge (CVPRW), 2018.

Publications not related to the topic of this thesis

1. Krešo, I., Šegvić, S., "Improving the Egomotion Estimation by Correcting the Calibration Bias", International Conference on Computer Vision Theory and Applications (VISAPP), 2015.
2. Krešo, I., Ševrović, M., Šegvić, S., "A novel georeferenced dataset for stereo visual odometry", Croatian Computer Vision Workshop (CCVW), 2013.

Životopis

Ivan Krešo rođen je u Sisku 1989. Magistrirao je računarsku znanost na Fakultetu elektrotehnike i računarstva Sveučilišta u Zagrebu 2013. godine. Nakon toga proveo je 6 godina na istom fakultetu zaposlen kao znanstveni novak na istraživačkim projektima te kao asistent u nastavi. Radno vrijeme provodi na istraživačkom radu te radu u nastavi. Sudjelovao je u nastanku predmeta Duboko učenje te radio kao asistent na predmetima iz područja arhitekture računala, oblikovnih obrzaca u programiranju i računalne grafike. Također je bio zaposlen na dva istraživačka projekta: *Multiclass Object Detection* (MULTICLOD) i *Computer Vision Innovations for Safe Traffic* (VISTA). Trenutno je zaposlen u industriji kao *data scientist* u RealNetworksu. Njegovi istraživački interesi uključuju klasifikaciju objekata, gustu predikciju za probleme semantičke segmentacije i detekcije, kao i samonadzirano i polunadzirano učenje.