University of Zagreb

FACULTY OF ELECTRICAL ENGINEERING AND COMPUTING

Marin Oršić

EFFICIENT SEMANTIC IMAGE SEGMENTATION USING PYRAMIDAL FUSION

DOCTORAL THESIS

Supervisor: Professor Siniša Šegvić, PhD

Zagreb, 2021.

University of Zagreb

FACULTY OF ELECTRICAL ENGINEERING AND COMPUTING

Marin Oršić

EFFICIENT SEMANTIC IMAGE SEGMENTATION USING PYRAMIDAL FUSION

DOCTORAL THESIS

Zagreb, 2021.

Sveučilište u Zagrebu

FAKULTET ELEKTROTEHNIKE I RAČUNARSTVA

Marin Oršić

UČINKOVITA SEMANTIČKA SEGMENTACIJA SLIKE PIRAMIDNOM FUZIJOM

DOKTORSKI RAD

Mentor: Prof. dr. sc. Siniša Šegvić

Zagreb, 2021.

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

Mentor: prof. dr. sc. Siniša Šegvić

Doctoral thesis contains: 82 pages

Doctoral thesis number.: ___________

About the Supervisor

Siniša Šegvic was born in 1971 in Split, Croatia. He completed elementary school and high school in Zadar, Croatia, with one year abroad in Milano, Italy (1985-86). He received the BS degree in electrical engineering (9 semesters) in 1996, from the Faculty of Electrical Engineering at the University of Zagreb, Croatia. From 1996 to 2005, he was employed at the Department of Electronics, Microelectronics, Computer and Intelligent Systems of the same Faculty, as a teaching assistant. He is currently employed at the same faculty as a full professor.

Siniša Šegvic participated in several national research projects (2 Croatian and 1 French), one Croatian national technology project and was a leader of another technology project. He was in charge of the research project MULTICLOD: Multiclass object detection (HrZZ, 2014-2017).

He received MS (2000) and PhD (2004) degrees in computer science from the University of Zagreb, Croatia. In 2005, he started a one-year postdoc position at IRISA, Rennes, France, in the field of appearance-based navigation by monocular computer vision. In 2006, he started a one-year postdoc position at TU Graz, Austria, in the field of monocular simultaneous localization and mapping, funded by a Marie Curie international incoming fellowship. His research and professional interests include 3D, active and distributed computer vision, especially in the context of localization and mapping for navigation purposes. He is also interested in image processing, software engineering, and generic and object oriented programming. He is the author or co-author of several papers published in international conference proceedings and reviewed scientific journals.

Siniša Šegvic 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. godine u Splitu. Osnovnu školu i matematičku gimnaziju završio je u Zadru, osim osmog razreda osnovne škole kojeg je pohađao u Milanu, Italija. Od lipnja 1996. godine do danas, zaposlen je na Zavodu za elektroniku, mikroelektroniku, računalne i inteligentne sustave Fakulteta elektrotehnike i računarstva u Zagrebu. U svibnju 2000. godine obranio je magistarski rad pod naslovom "Uporaba projekcijske geometrije i aktivnog vida u tumačenju scena”. Doktorsku disertaciju pod naslovom "Višeagentsko praćenje objekata aktivnim računalnim vidom" obranio je u lipnju 2004. U srpnju 2006. okončao je jednogodišnje postdoktorsko usavršavanje na institutu IRISA u Rennesu, Francuska, na području primjene račnalnog vida u samostalnoj navigaciji vozila u urbanom okruženju. U rujnu 2007. okončao je jednogodišnje postdoktorsko usavršavanje na tehnickom sveučilištu u Grazu, Austrija, na području analize nesigurnosti procjene geometrije dvaju pogleda.

Siniša Šegvić je sudjelovao u radu više domaćih i inozemnih znanstvenih projekata. Vodio je jedan istraživački projekt u suradnji s gospodarstvom (HrZZ, 2008-2011), jedan projekt primjene informacijske tehnologije (MZT, 2003-2004), te jedan razvojni projekt Sveučilišta u Zagrebu (2012). Bio je suvoditelj jednog bilateralnog austrijsko-hrvatskog projekta (MZOŠ, 2010-2012). Bio je voditelj istraživackog projekta MULTICLOD (HrZZ, 2014-2017). Njegovi znanstveni, istraživacki i profesionalni interesi uključuju računalni vid, obradu slike, programsko inženjerstvo te objektno i generičko programiranje. Samostalno odnosno kao koautor objavio je više članaka u časopisima s međunarodnom recenzijom te na međunarodnim znanstvenim skupovima. Kao recenzent, sudjelovao je u prosudbi članaka podnesenih za objavljivanje na znanstvenim skupovima i u časopisima.

Tijekom rada na Fakultetu elektrotehnike i računarstva, Siniša Šegvić je održavao predavanja na kolegijima Dinamička analiza scena, Oblikovni obrasci u programiranju, Arhitektura računala 2, Skriptni jezici, Inteligentni sustavi, Duboko učenje i Modeli za predstavljanje slike i videa. Za potrebe nastave, priredio je veći broj didaktičkih tekstova, koji su dostupni na mrežnim stranicama fakulteta. Suautor je knjige Python za znatiželjne. Konačno, sudjelovao je i u radu fakultetskog Odbora za istraživanje i međunarodnu suradnju.

Siniša Šegvić vrlo dobro poznaje engleski i talijanski, a služi se i francuskim jezikom. Član je strukovne udruge IEEE. Oženjen je i ima troje djece.

Abstract

Emergence of large datasets and resilience of convolutional models have enabled successful training of very large semantic segmentation models. However, high capacity implies high computational complexity and therefore hinders real-time operation. We therefore study compact architectures which aim at high accuracy in spite of modest capacity. We propose a novel semantic segmentation approach based on shared pyramidal representation and fusion of heterogeneous features along the upsampling path. The proposed pyramidal fusion approach is especially effective for dense inference in images with large scale variance due to strong regularization effects induced by feature sharing across the resolution pyramid. Interpretation of the decision process suggests that our approach succeeds by acting as a large ensemble of relatively simple models, as well as due to large receptive range and strong gradient flow towards early layers. We propose a novel semantic segmentation approach based on pyramidal representation with shared parameters and fusion of heterogeneous features along the upsampling path. The proposed pyramidal fusion approach is especially effective for dense inference in very large images due to very large receptive field of the resulting predictions. Validation and ablation experiments support our design choices and suggest that the proposed approach succeeds by acting as an ensemble of relatively simpler models. Our best model achieves 76.4% mIoU on Cityscapes test and runs in real time on low-power embedded devices. In this thesis, we describe the main components of a real-time semantic segmentation system based on deep convolutional models. We are considered with convolutional encoders used for recognition, as well as decoders which are crucial for obtaining accurate results. We do extensive evaluation of the developed method over a range of public and domestic datasets. Finally, we presents results in the 2020 instance of Robust Vision Challenge.

Keywords: semantic segmentation, real-time inference, shared resolution pyramid, computer vision, deep learning

Prošireni sažetak

Pojava velikih podatkovnih skupova i otpornost konvolucijskih modela omogućili su uspješno treniranje velikih modela za semantičku segmentaciju. Doduše, velik kapacitet modela zahtjeva visoku računsku složenost što onemogućuje primjene u stvarnom vremenu. Ova disertacija razmatra kompaktne arhitekture koje ganjaju visoku prediktivnu moć usprkos skromnom kapacitetu. Istraživanje je usmjereno prema konvolucijskim arhitekturama s optimiziranim konvolucijskim izvedbama sposobnim za klasifikaciju u velikim skupovima podataka. S ciljem ostvarivanja predikcija visoke razlučivosti razmatraju se efikasni blokovi za naduzorkovanje. Istraženi su pristupi za izlučivanje značajki iz piramide ulaznih slika s ciljem ostvarivanja invarijantnosti na skalu te uvećavanja receptivnog polja modela.

Ova disertacija bavi se semantičkom segmentacijom slike. Radi se o zadatku u računalnom vidu čiji je cilj razumijevanje slike na razini piksela. Ovaj zadatak pripada grupi problema guste predikcije: sustav mora pikselu slike na ulazu pridružiti semantički razred. Trenutno se najbolji modeli za semantičku segmentaciju oslanjaju na arhitekture s golemim računskim budžetima. Takve arhitekture nisu prikladne za primjene koje zahtjevaju izvedbu u stvarnom vremenu na ugradbenim računalima niske potrošnje odnosno u okruženju skromnih računskih resursa. Mnogi dosadašnji radovi koji ostvaruju efikasnu izvedbu koriste prilagođene arhitekture koje nisu prikladne za raspoznavanje u velikim skupovima podataka. Često ovakvi pristupi pokazuju sklonosti za prenaučenost. Ovo istraživanje je pokazalo kako su pozitivni efekti predtreniranja manji kod prilagođenih arhitektura.

Glavni doprinos ove disertacije je arhitektura dubokog modela za semantičku segmentaciju slike temeljena na obradi rezolucijske piramide. Obrada rezolucijske piramide redovito je korištena u klasičnom računalnom vidu. Međutim, pojavom dubokog učenja rijetko pronalazimo takve pristupe. Operacija konvolucije nije ekvivarijantna na promjenu skale, što znači da modeli temeljeni na konvolucijskim slojevima moraju trošiti kapacitet kako bi omogućili točno raspoznavanje preko različitih skala. Dvije ključne osobine obrade piramida slika su računska efikasnost na malim rezolucijama te ušteda kapaciteta ostvarena dijeljenjem parametara preko svih elemenata piramide. Ova disertacija predlaže piramidnu fuziju: arhitekturu dubokog modela za obradu rezolucijske piramide dijeljenim parametrima. Pokazuje se kako takav pristup ostvaruje najveću točnost među svim metodama čija primjena zahtjeva izvedbu u stvarnom vremenu. Nadalje, pokazuje se kako je premoć ovog pristupa veća u zahtjevnijim podatkovnim skupovima koji pobliže oslikavaju stvarne primjene algoritama za vizualno raspoznavanje.

Ranija evaluacija modela piramidne fuzije pokazala je pretreniranje na najgrublju rezoluciju. Također, učenje modela s povećanim brojem elemenata slikovne piramide vodilo je prema lošijim rezultatima. Uvođenjem prilagođene funkcije gubitka uspješno su riješena oba navedena problema. Ova funkcija gubitka modulira iznos gubitka negativne log izglednosti na način

da povećava gubitak u pikselima koji se nalaze blizu ruba dva semantička razreda. Dodatno, funkcija naglašava gubitak u pikselima za koje je model dodijelio malu vjerojatnost točnog razreda.

Razvijeni postupci su evaluirani na javno dostupnim podatkovnim skupovima za semantičku segmentaciju. Analiza rezultata korisna je istraživačima koji se bave efikasnim izvedbama s obzirom na temeljitost evaluacije, raznolikost podatkovnih skupova te validacijske eksperimente. Osim metrika točnosti, prikazane su i teoretske složenosti modela. Prikazano je i empirijsko mjerenje vremenskog izvođenja na hardverima različitih karakteristika. Konačno, programske komponente su javno objavljene s ciljem omogućavanja daljnjeg razvoja i analize postupaka predloženih u okviru ove disertacije.

Pregled poglavlja

Uvodno poglavlje prikazuje česte probleme s kojima se susreću današnji modeli za semantičku segmentaciju te pokazuje motivaciju za rješavanje ovog problema. Analiza računske složenosti postupaka pokazuje odnos među metodama koje ostvaruju visoku točnost i metoda koje podržavaju rad u stvarnom vremenu. Nadalje, objašnjena je intuicija iza korištenja rezolucijskih piramida za semantičku segmentaciju slike, što je podloga glavnog doprinosa ove disertacije.

Drugo poglavlje opisuje najznačajnije arhitekture modela za klasifikaciju slike u velikim skupovima podataka. Moduli klasifikacijskih modela dizajniranih za efikasno izvođenje korišteni su u predloženoj arhitekturi dubokog modela za efikasnu semantičku segmentaciju. Stoga je razumijevanje elemenata klasifikacijskih arhitektura preduvjet za uspješno razmatranje ostatka ove doktorske disertacije. Dijelovi naučenog klasifikacijskog modela mogu se iskoristiti za izgradnju segmentacijskog modela. Ovo se postiže izbacivanjem sloja globalnog sažimanja prosječnom vrijednosti te posljednjeg potpuno povezanog sloja. Često se ovako prilagođeni konvolucijski slojevi zovu koderima (smatra se da oni kodiraju semantičku informaciju). Pokazalo se kako korištenje klasifikacijskih kodera omogućuje efikasnu izvedbu visoke prediktivne moći.

Treće poglavlje opisuje značajne pristupe iz literature koji se bave semantičkom segmentacijom slika. Također, opisani su moduli korišteni za izgradnju efikasnih segmentacijskih modela. Nadalje, opisani su i postojeći pristupi za semantičku segmentaciju u stvarnom vremenu. Pristupe za efikasnu segmentaciju dijelimo u dvije glavne skupine. Prva skupina koristi prilagođene segmentacijske arhitekture, dok druga skupina modela koristi klasifikacijske kodere. Iz pregleda literature možemo zaključiti kako efikasna segmentacijska arhitektura mora sadržavati kodere viskog kapaciteta za ostvarenje visoke točnosti.

Četvrto poglavlje opisuje dvije predložene arhitekture konvolucijskih modela za semantičku segmentaciju. U početku poglavlja dan je pregled izvedbi konvolucijskih inačica. Zatim je opisana jednorazinska arhitektura koja koristi konvolucijski sloj prostornog piramidnog saži-

manja značajki. Ovaj sloj se koristi za povećanje receptivnog polja kodera čime se postiže ispravna segmentacija velikih objekata. Višerazinska arhitektura postiže sličan učinak akumulacijom značajki jednake prostorne rezolucije prilikom obrade slikovne piramide uz dijeljenje parametara kodera. Ovaj arhitekturni obrazac zovemo piramidnom fuzijom. Opisana je i nova formulacija gubitka koja naglašava doprinos iznosa gubitka u onim pikselima koji se nalaze bliže semantičkom rubu. Pokazuje se kako je nova funkcija gubitka ključan faktor u uspješnom učenju višerazinskog modela.

Peto poglavlje sadrži iscrpnu eksperimentalnu evaluaciju predloženih postupaka. Glavni rezultati uključuju usporedbu s postojećim stanjem stvari na javno dostupnim podatkovnim skupovima Cityscapes, CamVid, ADE20k i Mapillary Vistas. Zatim je pokazana mogućnost izvedbe u podatkovnim skupovima domaćeg podrijetla. Radi se o podatkovnim skupovima koji su prikupljeni u sklopu IRI projekta SafeTram, odnosno od strane tvrtke RoMb Technologies. U ostatku eksperimentalnog poglavlja opisan je postupak za mjerenje brzine izvođenja na grafičkim procesorima i na ugradbenim računalima niske potrošnje. Nadalje, prikazan je niz eksperimenata s ciljem objašnjavanja postupka zaključivanja razvijenog dubokog modela. Konačno, mjerenje efektivnog receptivnog polja pokazuje korisnost piramidne fuzije prilikom ostvarivanja modela za precizno i efikasno zaključivanje.

Šesto poglavlje opisuje rezultate na natjecanju u Robust Vision Challenge 2020. U natjecanju sudjeluju istraživačke skupine koje demonstriraju svoje metode u glavnim zadacima u računalnom vidu. Najveći naglasak natjecanja je otpornost modela na promjene okruženja. Robusnost se evaluira u višedomenskoj okolini: podatci za testiranje objedinjuju podatke iz više domena, čime ispitne slike imaju drastično različite karakteristike. Višedomenska evaluacija ističe mane onih metoda koje su prekomjerno prilagođene jednoj domeni. Nadalje, veličina podatkovnih skupova te broj razreda koje je potrebno prepoznati stvara velike zahtjeve za računskim resursima. U ovom poglavlju opisujemo primjenu piramidne fuzije u zadatku višedomenske semantičke segmentacije. Model piramidne fuzije postigao je najtočniji natjecateljski rezultat čime se naglašava značaj doprinosa prikazanih u ovoj disertaciji.

Sedmo i posljednje poglavlje donosi zaključke i osvrt na ovu disertaciju.

Ključne riječi: semantička segmentacija, zaključivanje u stvarnom vremenu, dijeljena rezolucijska piramida, računalni vid, duboko učenje

Contents

Chapter 1

Introduction

Architectural advances of deep models for image classification have immensely contributed to other visual recognition tasks. Modern approaches to object detection [1], instance segmentation [2], and semantic segmentation [3] yield best results with recent convolutional architectures. Convolutional models have also produced state-of-the-art results on visual reconstruction tasks such as stereo reconstruction [4, 5] and optical flow [6, 7, 8, 9, 10, 11].

This thesis is concerned with semantic segmentation which is also known as pixel-level image understanding. The task is posed as dense prediction: the system has to predict a semantic class for each image pixel. Compared to instance-level semantic segmentation which addresses only the "thing" classes [12], semantic segmentation is able to recognize the "stuff" classes as well. Panoptic segmentation [13] which was introduced only recently aims to recognize both "things" and "stuff" and enumerates "thing" instances. Currently, the best semantic segmentation accuracy is achieved with very large models which require extraordinary computational resources [14, 15, 16]. However, many important applications such as autonomous navigation or driver assistance require real-time inference on very large images in order to cover a wide field of view and perceive small objects at large distances. At the same time, these applications require very low latency in order to be able to bring real-time decisions. These opposing requirements intensify computational strain and make real-time implementations a challenging research objective.

Many real-time semantic segmentation approaches [17, 18, 19, 20] address this problem by introducing custom lightweight architectures which are not suited for large-scale visual recognition. Most of these approaches train from scratch and therefore miss a huge regularization opportunity offered by knowledge transfer [21] from larger and more diverse recognition datasets [22]. Consequently, these approaches incur a comparatively large overfitting risk. Some approaches alleviate this shortcoming by pre-training on ImageNet [17]. However, our experiments suggest that the resulting benefits tend to be smaller than in architectures aiming at competitive ImageNet performance.

Semantic segmentation models can be applied to many domains. Some possible applications are introduced through publically available datasets. A portion of these datasets is visualized in Figure 1.1. Application which attracts the most attention, and therefore most funding, is autonomous driving [8, 23, 24]. This application domain seeks to enable pixel-level understanding in driving scenes, where an autonomous vehicle would be equipped with one or more camera sensors. Data from the cameras would then be processed with a segmentation model to enable route planning which is supported by the semantics of the surrounding environment. Another interesting application focuses on segmenting buildings in aerial images [25]. Having pixelwise semantic understanding of aerial imagery would help automate processes in geodesy, where exist lots of inaccurate data gathered before the development of precise measuring devices. In agriculture, autonomous robots need to discriminate between crop types to successfully perform activities such as watering, trimming or weeding [26]. Finally, there are lots of robotics applications in household environments [27].

The simplest model for semantic segmentation would start with a fully convolutional encoder which gradually decreases the resolution and increases the number of feature maps of the resulting representation. Instead of performing global pooling (as in image-wide classification), one would proceed by attaching a pixel-wise loss to obtain the predictions [28]. This model would lead to very fast inference, however its accuracy would be rather low due to following two problems. Firstly, small objects (e.g. distant traffic signs) would likely be missed due to low output resolution, which is usually 32 times smaller than the input image. Secondly, the receptive field would not be large enough to recognize pixels at large objects (e.g. buses or trucks close to the camera). These problems can be alleviated with learned upsampling [3], dilated convolutions [29], lateral connections [30, 31, 32, 33, 34] and resolution pyramids [28]. However, not all of these techniques are equally suited for real-time operation.

This thesis presents a novel architecture for efficient dense prediction. The architecture extracts features at multiple levels of the resolution pyramid. Each level of the pyramid subsamples the resolution two times with respect to its predecessor. Hence, the upper bound for the computational overhead of the pyramid is $1/4 + 1/16 + 1/64 + \ldots = 1/3 \approx 33\%$. The feature extractor weights are shared across scales in order to promote feature reuse and reduce overfitting [28]. Cross-scale representations are assembled by adding features coming from different levels of the pyramid. This processing step is our main novelty and we denote it as pyramidal fusion. Pyramidal fusion results in a well-connected model which favours generalization due to acting as a large ensemble of independent simpler models. Additionally, it induces a large receptive range while requiring very little additional capacity with respect to the single-scale model. We complete the architecture with a lightweight ladder-style decoder [33], which gradually blends the fused representations along the upsampling path. This step is responsible for achieving high accuracy near semantic borders, together with boundary-aware loss [35].

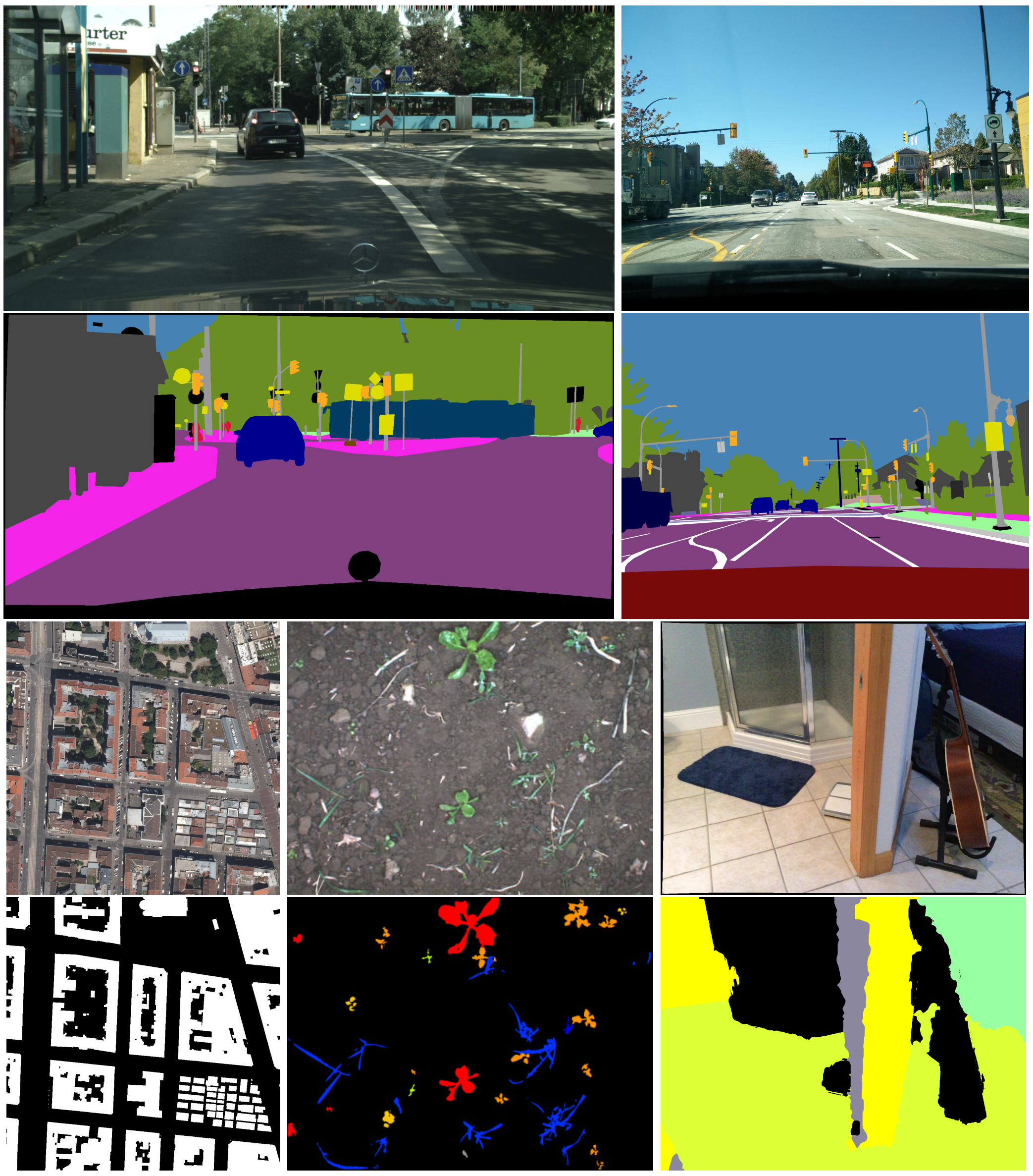

Figure 1.1: Image-label tuples from popular public datasets. The datasets are collected for different purposes and have widely different characteristics. Labels contain class indices for each pixel, which are color-coded for visualization purposes. Examples include tuples from: Cityscapes [23], Mapillary Vistas [24], INRIA aerial dataset [25], Sugar Beets dataset [26] and ScanNet [27].

Computer vision systems process images acquired by perspective projection of light rays to the sensor (image) plane. The nature of such projection implies that scale of the observed object is dictated by the object physical size and its distance from the image plane. An effective computer vision approach should not be brittle to variations in object size. Instead, a well formed approach would be invariant to changes in object scale. Furthermore, invariance to scale should not imply a large computational increase and, ideally, the method should not waste effective capacity by processing each scale variation separately. Obtaining scale invariant features is beneficial in all computer vision tasks. Scale invariant feature transform (SIFT) [36] extracts local features capable of handling variations in rotation and scale, making them suitable for robust feature matching. SIFT processes scale-space representations [37] across an image pyramid. The resulting features are efficient to compute, and offer traits suitable for sparse matching in a variety of reconstruction tasks, as well as setting ground to object recognition. Since the advent of deep learning, quality of recognition systems was improved repeatedly by enforcing scale invariance. Kreso et al. [38] consider using distance information to choose appropriate scales in the image pyramid to classify each pixel of the presented image. Singh and Davis [39] apply a region proposal network (RPN) to an image pyramid. The RPN is trained to propose objects only at a moderate scale, ignoring too large or too small objects (these have moderate size in a different pyramid level). The main contribution of this thesis is focused around observing objects in an image pyramid using a shared set of convolutional parameters. This is especially effective when aiming at efficient inference, as we demonstrate accurate semantic segmentation results across different settings while supporting real-time execution.

The idea behind using resolution pyramids in recognition tasks is visualized by the example in Figure 1.2. Let us consider the classification problem behind the red pixel on the large truck. The original image resolution is 2048×1024 pixels. Also, let the theoretical (maximal) receptive field of the model at hand be 200×200 pixels. This means that this convolutional model classifies the red pixel based on pixels captured within the 200×200 area only. The yellow rectangle in the top right image visualizes the receptive field at original input resolution. Yellow rectangles in bottom left and right images visualize the same 200×200 receptive field once the input image is 2 and 4 times subsampled, respectively. For clarity, Figure 1.3 visualizes these 200×200 pixel regions from each level of the resolution pyramid. The example illustrates how discriminative features useful for properly classifying the truck pixel, such as wheels or headlights, are possible only in the low resolution pyramid level.

We propose two compact architectures which deliver competitive recognition performance under real-time constraints. Both architectures are very suitable for low-cost embedded devices such as Jetson TX2 and Jetson Nano. We revisit feature reuse across the resolution pyramid – a prominent regularization technique which has been neglected in recent literature. We propose to accompany this approach with pyramidal fusion and boundary-aware loss, and demonstrate significant improvements on all tested datasets. especially when training from scratch. Finally, we present a series of experiments which interpret the decision process of our models. First, we perform an attribution study which quantifies the success of capturing a wide context of pixel-level predictions, and shows its correlation with generalization performance. Second, we explain fast learning and improved generalization of pyramidal fusion by improved gradient flow and ensemble-like behaviour. Third, we illustrate importance of ImageNet pretraining by showing that it affects generalization performance of all four processing blocks of the backbone.

In comparison with our preliminary report [40], our more recent publication [41] proposed an improved training objective and introduced several other enhancements which together de-

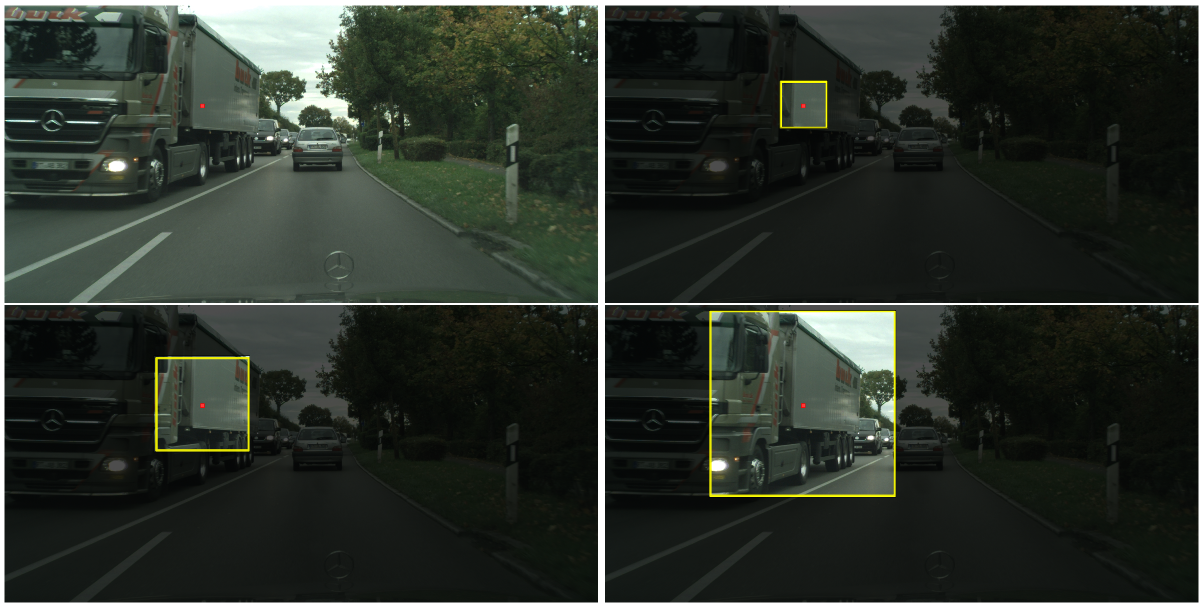

Figure 1.2: Top left: a typical input image presented at the model input. Classification of the pixel marked in red is considered for a model with receptive field 200×200 pixels. The receptive field is visualized in the original (top right), two (bottom left) and four (bottom right) times subsampled images.

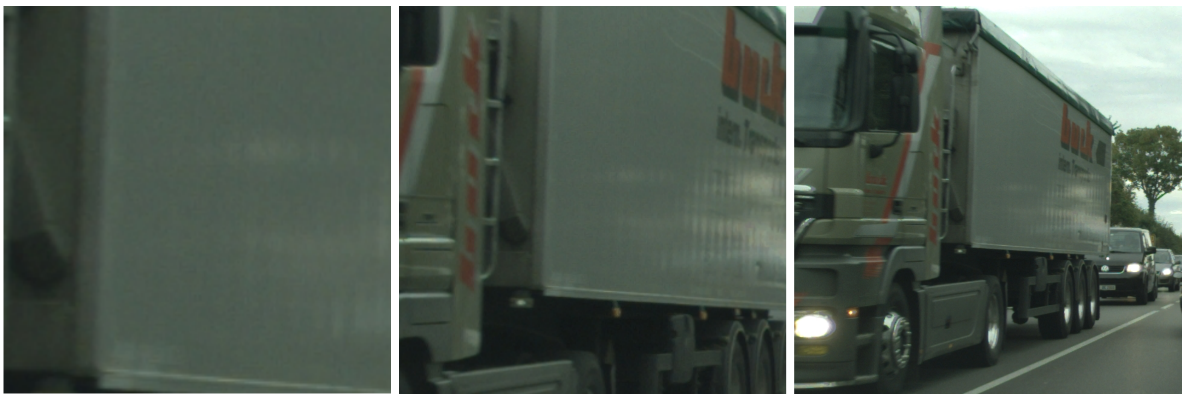

Figure 1.3: Pixels observed by a limited receptive field model across the resolution pyramid when inferring the class of the central pixel. Images (from left to right) are 1x, 2x and 4x subsampled.

liver state-of-the-art real-time performance on Cityscapes test. Additionally, we included experiments which evaluated variants of spatial pyramid pooling (SPP) [14, 15], validated the impact of boundary-aware loss, measured execution speed on Jetson Nano and RTX 2080 Ti, and evaluated semantic segmentation accuracy on Vistas and ADE20k. Finally, we presented experiments which interpret what our models have learned. In particular, we showed that pyramidal fusion has a large effective receptive field, which is required for correct recognition of locally indistinctive regions. We presented experiments which suggest that a dense prediction model with pyramidal fusion acts as an ensemble of smaller models, unlike its single-scale counterpart. We also showed that ImageNet pretraining significantly affects generalization performance of all backbone modules. This thesis consolidates the contributions from our previous publications. Furthermore, it presents an overview of ImageNet classification architectures, which is important for understanding encoders used in semantic segmentation models. Also, this thesis includes results on two datasets collected in Croatia. Finally, we present the winning submission to Robust Vision Challenge 2020 which uses the proposed architecture in multi-domain semantic segmentation. These results demonstrate how pyramidal fusion may be useful in applications requiring large capacity, like training on almost 200 semantic classes across seven different domains.

The proposed architecture is very well suited for poorly balanced semantic segmentation datasets with large resolution and large objects. It can deliver high recognition accuracy even when configured with a recognition backbone with comparatively low capacity. In particular, we consider lightweight ImageNet-grade architectures [42, 43] in order to achieve efficient inference and benefit from transfer learning. In a nutshell, we compensate the backbone capacity with increased receptive field, scale covariance due to shared parameters, and ensembling effect due to pyramidal fusion. The resulting architecture is suitable for real-time operation even on embedded GPU platforms. Our models achieve state-of-the art semantic segmentation accuracy among all existing approaches aiming at real-time execution.

The contents of this doctoral thesis are organized as follows. First we introduce architectural advances of deep convolutional architectures for image classification in large datasets. This overview helps better understand concepts which lead to accurate and efficient training of contemporary models for image understanding. Next, we present advances in the literature within the semantic segmentation task. Main focus of the analysis are models capable of real-time inference. Afterwards, we introduce our two approaches to real-time semantic segmentation. Here, we describe the main building blocks of the architectures as well as the novel boundary aware loss function used for training our pyramidal fusion model. Following the method overview, we present experiments on public datasets as well as ablation studies and explainability experiments. We describe a framework for measuring inference speed and present measurements on a range of hardware components. We also demonstrate results on two locally collected datasets, which study in-the-wild applicability of our method. Finally, we present an overview of our submission to Robust Vision Challenge 2020.

We outline the main contributions of this doctoral thesis. We presents a residual convolutional architecture for real time semantic segmentation. Next, we introduce pyramidal fusion: an architectural element for dense prediction with interleaved upsampling of shared resolution pyramid features. The pyramidal architecture is improved by a novel loss function which prevents overfitting to the coarsest level. We release software components which enable efficient inference of the proposed methods on graphics processing units. Finally, we perform a thorough analysis of our methods on public datasets and compare them to recent advances from the literature.

Chapter 2

Deep convolutional architectures for image classification

Image classification is considered to be a primary image recognition task in computer vision [44]. The typical classification task includes N images and a label set consisting of C classes. A classification model assigns a single label to the entire image. Before the advent of deep learning this challenging task has been addressed by approaches based on hand-crafted features such as SIFTs [36, 45] or Fisher vectors [46]. These approaches will not be considered here. Instead, this chapter will focus on image classification approaches based on deep learning [42, 44, 47]. Most of these architectures were designed for image classification on the ImageNet dataset [22]. This dataset consists of over one million labelled images split into one thousand classes which cover a vast amount of visual concepts present in human surroundings. When addressing image classification, the research community used to consider results on MNIST [48], SVHN [49] or CIFAR [50]. Nowadays these are considered as toy datasets. On the other hand, ImageNet is still the main benchmark when comparing image classification architectures. More recently, larger datasets such as OpenImages [51] emerged but only a handful of research groups have sufficient computing power to perform experiments on such scale. Models designed for ImageNet are applicable for other datasets and tasks. Therefore, understanding the principles used for designing these models are important for every computer vision practitioner.

Deep classification architectures have also been used in other recognition tasks. A deep model trained on ImageNet may be viewed as a nonlinear feature extractor complemented with a jointly trained classifier based on multi-class logistic regression. Such feature extractors and trained parameters may be used as a starting point in solving a different task. Some of these tasks are: semantic segmentation [28], object detection [52], instance [2], panoptic segmentation [13] etc. This broad applicability in different tasks serves as motivation for studying image classification architectures. This thesis employs the resulting feature extractors for efficient

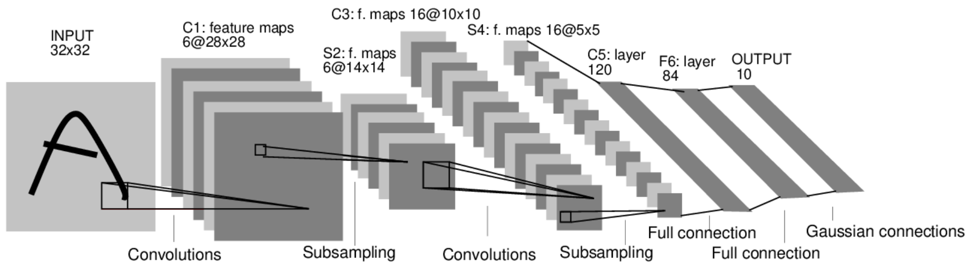

Figure 2.1: The LeNet-5 classification architecture used for classification of handwritten digits. The figure is reproduced from [48].

semantic segmentation.

2.1 LeNet, AlexNet

This section gives an overview of historically significant architectures: LeNet and AlexNet. Although modern classification models yield significantly higher accuracy, it should be noted how some fundamental concepts did not change. Feature extraction is realized using convolution operations, where convolutional kernels are found by minimizing a loss function. A deep model is defined as a mapping between the input and it's output which is composed of linear projections and nonlinear activations. LeNet and AlexNet are considered "deep" because nonlinear scalar functions are placed between linear convolutional operations.

2.1.1 LeNet

The LeNet model successfully addresses the classification task of handwritten MNIST digits [48]. This work is considered as the first modern convolutional architecture. Convolutions are a good fit for image recognition due to their translation equivariance [*]. Another earlier approach addressed a similar problem but with less attention in the research community [53]. While LeNet has large differences compared to latter described models, the core principle is the same: feature extraction is based on a set of convolution operations with parameters trained using the backpropagation algorithm [54]. Figure 2.1 depicts the architecture of LeNet-5. Between each linear operator there is a nonlinear activation. In the LeNet architecture, this nonlinearity is a sigmoid. Invariance to local displacements is alleviated by summation pooling inside a 2×2 window which is applied after convolution. Trainable bias and scaling is applied to pooled features. After the last convolution there are two fully connected layers which are expressed by matrix multiplication and nonlinear activation.

In comparison with deep models, the main difference is the definition of the loss function. Each of the C LeNet's outputs is expressed as the vector norm of the difference between ex-

[*]Convolutions are not equivariant to changes in scale. This thesis proposes a method to alleviate this issue.

tracted features **x** and untrainable center **W**:

$$y_i = \sum_j (x_j - w_{ij})^2 = \|\mathbf{x} - \mathbf{w_i}\| \tag{2.1}$$

The best overlap between the feature and centre vector occurs when the two vectors are the same i.e. when the vector norm of their difference is zero. Classification is made by finding the smallest component in the output vector:

$$c = \arg\min \mathbf{y}. \tag{2.2}$$

The loss function suitable for this output interpretation is mean squared error (MSE) with an added term which minimizes the posterior probability of incorrect classes:

$$E(\hat{y}, \mathbf{y}) = y_{\hat{y}}^2 + log(e^{-j} + \sum_{i!=\hat{y}} e^{-y_i}). \tag{2.3}$$

Here, $\hat{y}$ is the index of the correct class and **y** is the model output. It should be noted how the minimum ot the MSE loss component leads to the trivial solution where all outputs equate to zero. Therefore, the second loss component penalizes small values of the output in incorrect classes.

The usual loss function for training deep classifiers is the negative log likelihood of the correct class. Due to a modified loss function, the classification decision is interpreted as the index of the maximum output vector component. Following the loss formulation, the model output is modelled using the $softmax$ function. Softmax is a function which converts the vector with C dimensions, commonly called logits, into a vector with C probabilities (the sum of its elements is one). The c-th output vector component is expressed by Eq. 2.4. **W** is the parameter matrix of the final fully connected layer, and **x** is the feature vector being classified.

$$\mathbf{y}(\mathbf{Wx})_c = softmax(\mathbf{Wx})_c = \frac{e^{\mathbf{W}_c\mathbf{x}}}{\sum_{j=1}^{C} e^{\mathbf{W}_j\mathbf{x}}} \tag{2.4}$$

2.1.2 AlexNet

The AlexNet architecture marks an important milestone in image classification [44]. It is the first deep model to outperform shallow learning in ImageNet image classification [22]. Furthermore, AlexNet outperformed shallow models by a wide margin. Some of the contributions are enumerated below.

ReLU activation function is used as the model nonlinearity [55]. The activation function is expressed as:

$$ReLU(x) = max(0,x) \tag{2.5}$$

where x is a scalar input. This activation function holds some important properties which enable efficient training. First, the upper bound of the function codomain is $+\infty$ i.e. it is unbounded from above. This property leads to gradients having no saliency areas (which occurs with sigmoid or hyperbolic tangent). For each positive input to $ReLU$ there exist nonzero gradients.

Local response normalization promotes model generalization by reducing activation scale. As mentioned before, ReLU is not upper bounded which enables situations where activations may tend to infinity. The normalization is defined by:

$$b^i_{x,y} = \frac{a^i_{x,y}}{(k + \alpha \sum_{j=max(0,i-n/2)}^{min(N-1,i+n/2)} (a^i_{x,y})^2)^\beta} \tag{2.6}$$

where i is the feature map index, (x,y) are spatial coordinates and n is the neighbourhood size used in computing the normalization. Note that local response normalization is not used nowadays as it was replaced by more powerful batch normalization [56].

In addition to highlighted contributions, a set of methods for achieving higher test accuracy were used:

- Training on multiple GPUs at once enables for a larger batch size, leading to more stable training,
- Overlapping regions in pooling layers: the kernel size in pooling layers with stride 2 is set to 3 (the usual setting is 2)
- Dropout, a regularization technique which approximates training multiple models and ensembling their outputs at test time [57].

The model architecture is visualized in Figure 2.2. The input image size is set to 224×224 and the model may not be applied to any other input size. This is due to fully connected layers at the model output depending on the input image size. In other words, parameter matrix dimensionalities of mentioned layers are determined by the input size. More recent work found how this limitation may be circumvented. The model architecture is composed of sequential convolution, ReLU, local response normalization and max pooling operations. The convolutional representation is rearranged into a vector suitable for vector-matrix multiplication inside fully connected layers.

The total parameter count is roughly 60 million. An involved reader will find the parameter placement analysis interesting. The final convolutional tensor dimensionality is 6×6×128. Flattening this representation produces a 9216 component vector which equals the row count

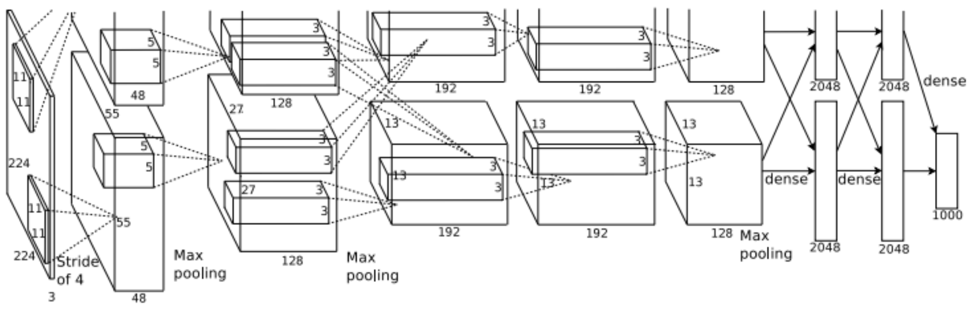

Figure 2.2: AlexNet convolutional architecture. Credit: Krizhevsky et al. [44].

in the first fully connected layer parameter matrix. This matrix has 4096 columns i.e. it's size is 9216×4096. The following fully connected layer produces a feature vector of the same size, therefore there is a 4096×4096 dimensional parameter matrix. The final matrix multiplication produces a vector which contains a scalar element for each dataset class. On ImageNet, a 1000-dimensional vector is produced at the model output. Finally, the total number of parameters inside fully connected layers equals to $9216*4096+4096*4096+4096*1000 = 58621952$. This analysis shows how almost 98% of all AlexNet parameters come from fully connected layers. This precludes constructing deeper models due to high tendency towards overfitting.

2.2 VGG

The VGG model has made a large impact in the research community [47]. The authors demonstrated how an increase in classification accuracy can be obtained by adding more convolutional units (and nonlinearities), thus making the model more discriminative. An explosion in parameter count caused by adding convolutional layers is alleviated by using smaller convolutional kernels. Earlier models commonly used 7×7 or 11×11 convolutions. Instead, VGG uses 3×3 convolutions. The reasoning behind this is as follows. A receptive field of three consecutive 3×3 convolutions is equal to using one 7×7 convolution. The advantage of using three sequential convolutions is a smaller parameter count and increased discriminativity induced by 2 extra nonlinear activations. Reduced computational complexity and implementation efficiency are additional advantages of using 3×3 convolutions [58]. Compared to im2col, the Winograd minimal filtering algorithm yields best performance increase when using 3×3 sized kernels. The cuDNN library uses the Winograd algorithm to speed up 3×3 convolutions.

When considering model organization, convolutional layers may be grouped by spatial resolution they operate at. In VGG, there are max pooling layers between convolutional groups, which halve the representation width and height. Usually, each subsequent convolutional group processes twice the feature maps and 4 times less spatial data. Consequently, each feature tensor uses half the memory when compared to tensors produced by the previous convolutional group

[†].

The family of VGG models starts with the VGG-A architecture, and continues from VGG-B up to VGG-E. The VGG-A model consists of eight convolutional and three fully connected layers. VGG-A parameters are trained from random initialization by sampling parameters from a zero mean and 10^{-2} variance normal distribution. Bias parameters are initially set to zero. All models larger than VGG-A are initialized in two steps. First, all parameters present in the nearest smaller model are initialized from there. Second, newly introduced parameters are initialized by aforementioned random sampling.

2.3 Architectures with skip connections

So far we considered models in which every output tensor is an input to one other function, exclusively. From the symbolic computational graph perspective, models considered up to this point consist of nodes connected by a single edge[‡]. Great advances in deep learning architectures were achieved by introducing skip connections. Here, an output of a particular node may be an input to multiple other nodes, as demonstrated in Figure 2.3. This results in models which better propagate loss gradients and consequently achieve better deep representations which yield best results, faster training and better generalization. This section considers ResNet and DenseNet architectures.

2.3.1 ResNet

Skip connections are considered as standard practice in contemporary deep architectures. Residual architectures (ResNets) are one of the first representatives of this concept [42]. Another concurrent work to ResNets, Highway Networks, demonstrated a similar concept [60, 61]. The idea is simply visualized by the computation graph in Figure 2.3. The input $\mathbf{x}$ is processed by operation $f_{k-1}(\mathbf{x}) = \mathbf{y}$ whose output is processed using $f_k(\mathbf{y})$. When considering models applied to images, these operations are usually sequences of convolution, nonlinearity and batch normalization. After applying non linear transformations to $\mathbf{x}$, the result is combined with the input by addition: $\mathbf{z} = \mathbf{x} + f_k(\mathbf{y})$. This defines a general form of a residual unit. By stacking resiudal units of common resolution, a residual block is formed.

Typical ResNet architecture can be described as follows. As with architectures described in sections 2.1 and 2.2, this model may also be considered a deep feature extractor complemented

[†]It was presumed that deep models achieve excellent generalization due to compressing information. However, this is not the case as there are high accuracy models which are bijective with respect to their input [59].

[‡]Modern frameworks for automatic differentiation like PyTorch or Tensorflow express models as computational graphs. This paradigm is suitable for runtime optimization of graphs independent of the execution platform. Furthermore, the backpropagation algorithm is defined directly by the computational graph itself, where each node is capable of performing forward and backward passes.

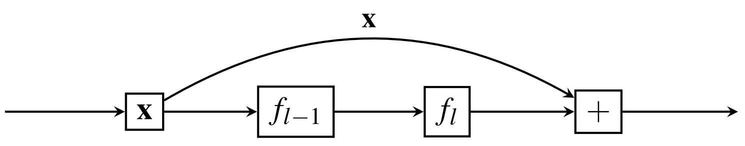

Figure 2.3: A residual unit displayed using a computational graph. Taken from [62]

with a multi-class logistic regression classifier. The two components are jointly trained from trained end to end. Among all models models described so far, ResNets contain all feature extractor parameters inside convolutions. Fully connected layers are not used here for two main reasons: i) they require a fixed input image size and ii) they contain the majority of model parameters. The deep feature extractor consists of the initial convolution ("stem") and four processing blocks which group convolutional units operating on the same resolution. The initial convolution operates on the input RGB image. It has a 7x7 kernel which is applied at stride two. Each processing block is preceded by a pooling layer with stride two which makes the output two times smaller than the block input. This first part is referred to as a stem. Following the stem, there are four residual blocks. Each block outputs a 2 times subsampled representation when compared to its input. The resulting features are 32 times subsampled with respect to the input image. Each residual block consists of sequential residual units which may be described using Figure 2.3. More specific, a residual function is a 3×3 convolution – batch normalization – ReLU composition [§].

Another significant novelty in the ResNet architecture is present at the end of the feature extractor. There, the global average pooling operation is applied. This operation pools the final residual block representation by taking the average values of all spatial locations in the feature map. The average value is taken for each feature map separately. Global average pooling is suitable for two main purposes:

- linear classifier at the model output is not dependent of the input image dimensionality, and
- the number of non-convolutional parameters is significantly reduced.

Particular attention should be paid to reduction in parameter count. Let us once again consider the ratio of parameters contained in convolutional and classifier layers of the baseline ResNet-18. This model has 512 features in the final convolutional representation which are classified in 1000 classes. Roughly, there are 512 thousand parameters in the linear classifier which is only 5% of all parameters. We see how ResNets tend to keep most parameters in convolutional layers which was not the case with older deep architectures. Insisting on the expressive power of convolutional layers makes sense due to the bias of deep convolutional models: image represen-

[§]Note that the operator ordering is not arbitrary. The distribution of outputs from a batch normalization layer follows a $\mathcal{N}(0,1)$ distribution. As already mentioned, ReLU has non zero gradient for positive values only. Therefore, the expected ratio of propagated gradients with respect to ReLU inputs is 50%.

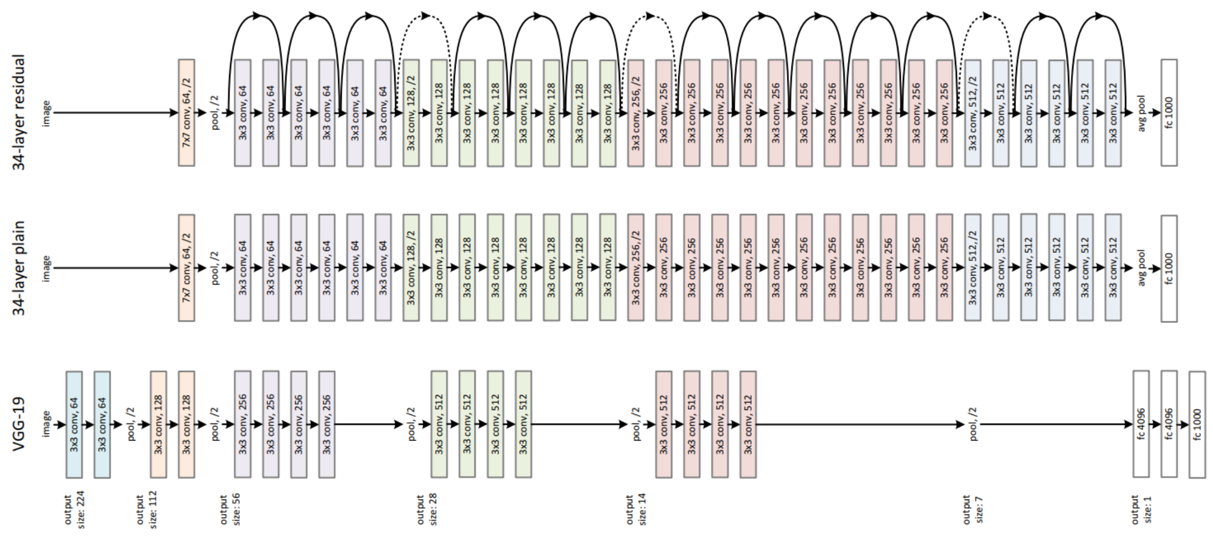

Figure 2.4: The ResNet classification architecture. Credit: [42].

tations have a hierarchical structure which is well described by a composition of convolutional layers.

Although a simple modification, the residual structure brings tremendous benefits to training deep models. As demonstrated by Simonyan and Zisserman [47], models without skip connections can only be trained to moderate depth. VGG-A, which has 11 layers, is the only model trained from random initialization. On the other hand, ResNet-152 is trained from scratch while achieving much higher classification accuracy than VGG-E.

Figure 2.4 visualizes two popular ResNet architectures. Arks with full arrows displays identity mappings, while discontinuous lines contain a transformation which adapts the tensor dimensionality. The following interpretation hypothesizes why it is possible to train such deep residual models. Let us find the shortest path between the input and the output. The shortest path is defined as the minimum number of non linear transformations. The shortest path contains:

- a 7×7 stem convolution,
- three 3×3 convolutions which adapt feature dimensionality of the residual block output, and
- a fully connected layer which outputs the final classification.

We see that the "shallowest submodel" is only 5 layers deep. A recent detailed analysis indicates how residual models act operate as ensembles of moderate depth models [63]. Furthermore, better generalization is obtained by initially directing the gradients towards the shallowest submodel. This is done by setting the parameters of final residual units to have zero valued outputs [64]. Using this initialization, the residual model is gradually becoming deeper as training progresses.

2.3.2 DenseNet

The use of skip connections demonstrated great qualities when building deep models. Node connectivity inside a computational graph is an important trait of the model structure primarily for promoting gradient propagation during model training. A model with L layers without any skip connections has L connections. Residual models can have a maximum of $2L$ connections[¶]. The DenseNet architecture introduces the concept of dense connectivity [65]. DenseNets contain $\frac{L(L+1)}{2}$ connections inside a convolutional group. A comparison between residual and dense connectivity is visualized in Figure 2.5. A densely connected block (a convolutional group operating at common resolution) consists of dense layers. Here, the l-th dense layer is directly connected to all preceding layers. This connectivity pattern is achieved using feature map concatenation: the input to the l-th dense layer is a concatenation of outputs from layers 0 to $l-1$.

It should be noted how each dense layer produces a small number of feature maps (typically 32). To keep computation and parameter count reasonable, a 1×1 convolution is applied to each dense layer input. Afterward, a 3×3 convolution is applied. A dense layer has the opportunity to filter unnecessary features using the 1×1 projection, although all preceding features are visible. Contrary, a residual model does not demonstrate this property as addition forces the use of all features.

Dense connectivity enables representations which do not suffer from vanishing or exploding gradients. Furthermore, it promotes feature reuse, increases parameter efficiency and requires less computation. Finally, a memory efficient implementation [65, 66] is a great advantage of this architecture which is interesting in memory intensive applications, like dense prediction on very large images or video analysis.

Important milestones in image classification advancement are outlined in Figure 2.6. The first breakthrough is the emergence of deep learning, more specifically AlexNet. The next big landmark occurred by introducing skip connections in deep models. This enabled deep models to surpass human-level accuracy in image recognition. Nowadays, advancement in classification accuracy is gradually saturating. The author sees three directions for advancement in image classification architectures. The first direction is reducing computational complexity while maintaining predictive power. The next research area focuses on models trained in unsupervised or semi-supervised settings [67, 68]. This settings focuses in developing methods which have satisfactory results when using little labelled data. Finally, scaling the datasets and the number of classes is a research area aiming at training models capable of successful generalization across drastically different environments [69].

[¶]In practice, ResNets have $\frac{3L}{2}$ connections.

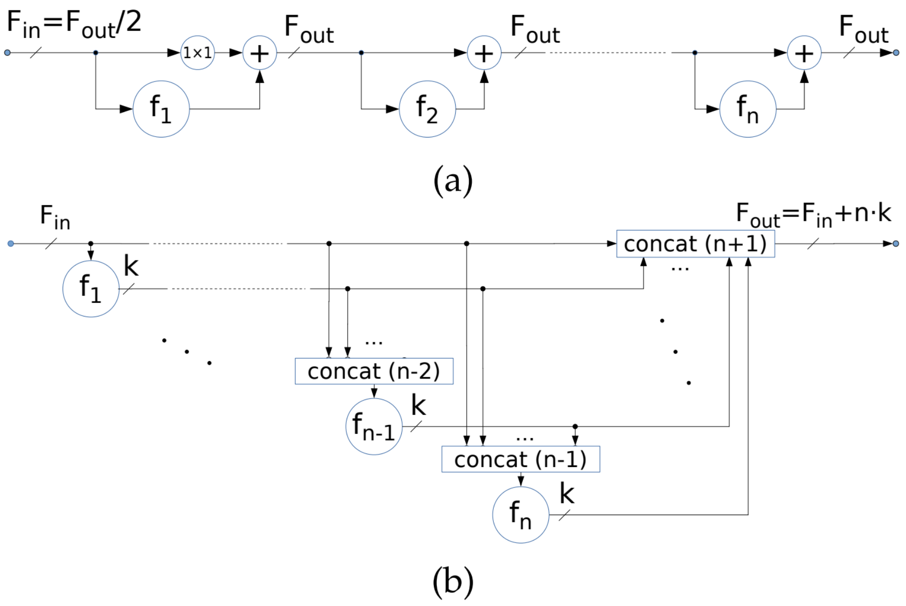

Figure 2.5: Comparison of layer connectivity in a) ResNet models and b) DenseNet models. Credit: [66].

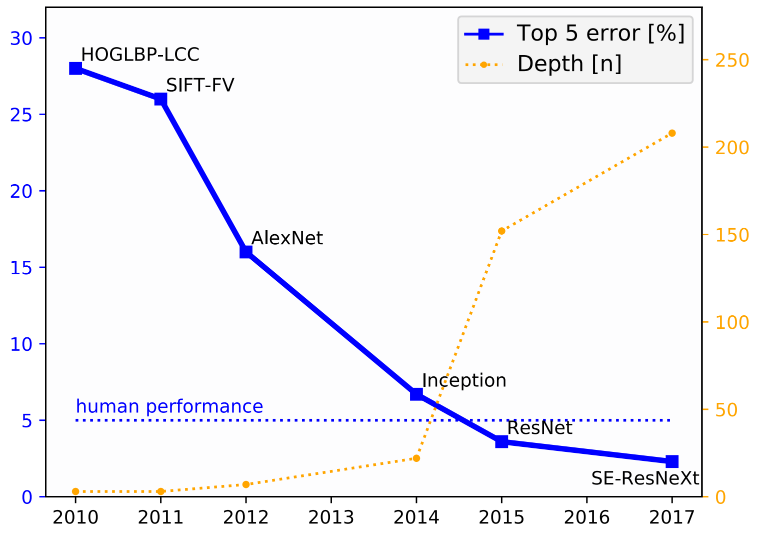

Figure 2.6: Advancement in ImageNet classification accuracy. The left ordinate shows the classification error on ImageNet validation subset. The right ordinate shows the model depth. Credit: [70].

2.4 Efficient convolutional architectures

Efficient convolutional layers reduce computational requirements of the standard convolution while aiming to keep good generalization. These techniques are detailed in subsection 4.1.1. Here we describe convolutional architectures which enable efficient classification by using optimized convolutional instances.

ShuffleNet [71] uses channel shuffling in combination with grouped convolutions to share information across different groups. CondenseNet [72] uses a training method which finds important connections and eliminates uninformative weights to reduce computation. Neural architecture search [73] (NAS) is a recent method which utilizes reinforcement learning to simultaneously find the model architecture and it's parameters. This approach finds accurate models given a computational budget. Therefore, an efficient model may be produced using NAS.

2.4.1 MobileNet V2

The second MobileNet architecture is an important milestone in developing deep architectures for efficient applications [43]. When considering computational complexity, MobileNet V2 is five times more efficient than the smallest ResNet while achieving the same accuracy. Furthermore, it has an around ten times less parameters compared to ResNets and a hundred times less parameters compared to VGG. This section gives an overview of MobileNet V2.

The main contribution is a novel convolutional block called inverted residual. As the name suggests, the convolutional block uses residual connectivity. Skip connections have negligible computational complexity which makes them suitable for efficient applications. Figure 2.7 compares a residual unit (a) to an inverted residual (b). An obvious similarity is the use of skip connections. However, there are two important differences:

- a standard 3×3 convolution is replaced by a depthwise separable instance, and
- the inverted residual increases the feature map count. Note that the shallowest submodel (one not passed through inverted residuals) contains a small number of feature maps.

An inverted residual begins with a linear projection to a higher dimensional space. The projection is implemented using 1×1 convolution. Higher dimensional features are then processed with depthwise separable convolutions. Finally, the inverted residual output is obtained using a linear projection to lower dimensional space which matches the size of the input tensor. Processing at a higher dimensionality inside the inverted residual block is feasible since an efficient convolutional instance is used.

2.4.2 EfficientNet

Model scaling is a term used to describe methods which take a small model and amplify its components to obtain a bigger model with higher accuracy. Naturally, there are lots of different methods for model scaling. Typical model hyperparameters used to scale a model are the number of layers (depth), the number of convolutional feature maps (width) and the input image resolution. Usually, more accurate models are obtained by scaling along one of these dimensions. The EfficientNet architecture demonstrates how best results are achieved when scaling all these components simultaneously [74]. This method for model scaling is named compound scaling. Compound scaling obtains hyperparameters of the larger model by multiplying each

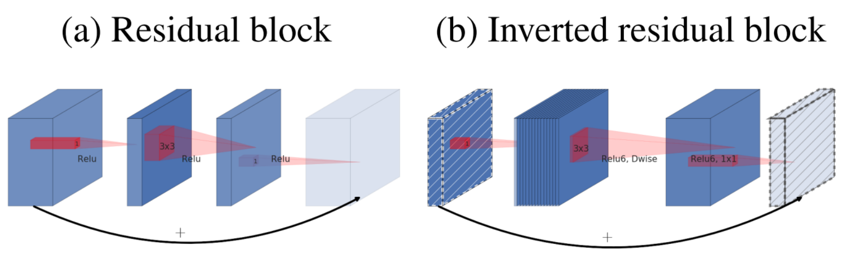

Figure 2.7: Comparison between a residual and an inverted residual block. Reference: [43].

dimension with a fixed factor. The resulting architectures surpass accuracy of existing models across different computational budgets.

The basic (smallest) architecture is retrieved using neural architecture search. The search resulted in a model using inverted residual blocks from MobileNet V2 [43] with different depth and width. This model is more efficient and has better accuracy than MobileNet V2. This basic architecture, named EfficientNet-B0, is expanded using compound scaling to create architectures from EfficientNet-B1 up to EfficientNet-B7. However, unlike VGG models, where each model uses trained parameters from the smaller instance, each EfficientNet is trained from random initialization. This is possible due to the model residual structure.

Finally, a few words should be said about the insights on MobileNet V2 and EfficientNet found in the scope of these thesis. Inference on general purpose CPUs and mobile device is indeed faster when using efficient convolutional blocks. However, training and evaluating deep models is always performed on high-end GPUs. It turns out that a small theoretical complexity of these models does not correlate with high inference speed when running them on modern GPUs. A basic analysis indicates how depthwise separable convolutions utilize cache memory worse than full 3×3 convolutions. However, a detailed analysis of these shortcomings is not in scope of this thesis.

Chapter 3

Convolutional architectures for semantic segmentation

3.1 Elements of efficient convolutional models

As described in the introduction, semantic segmentation models have to face two major problems: restoring the input resolution and increasing the receptive field. The simplest way to restore input resolution is to avoid downscaling. This can be achieved by replacing stride-2 layers with their non-strided counterparts, and doubling the dilation factor in subsequent convolutions. PSPNet [15], DeepLab [14, 75] and other such approaches perform this trick in the last two residual blocks of the recognition backbone. Hence, all image representations of these models are at most 8 times subsampled. However, this significantly increases the computational complexity and therefore precludes real-time inference on large images [66].

Another way to restore the input resolution relies on trained upsampling [3], which leads to the encoder-decoder architecture. The idea is to perform recognition on low resolution features (encoder), and then to restore details by upsampling the resulting representation (decoder). Many approaches follow the SegNet design [76] where all decoder layers and maps are in symmetry with the encoder. Trained upsampling can be naturally augmented with lateral (also known as ladder-syle) connections [30] in order to blend semantically rich deep features with spatially rich shallow ones. FCN [3] performs the blending by adding scores of upsampled features from different convolutional layers. However, this results in slow inference due to high-dimensional features at the output resolution, and poor accuracy due to simplicity of independent scoring. U-Net [31] proposes a symmetric encoder-decoder architecture, where the decoder is designed as a series of recurrent upsampling modules. Each upsampling module concatenates the upsampled previous representation with the lateral connection, and blends the two with a processing block whose complexity reflects the corresponding processing block in the encoder. However, we believe that recognition requires more capacity than locating borders when rough semantic content is known. This assumption is supported by empirical advantage of asymmetric encoder-decoder architectures in semantic segmentation [33] and object detection [32]. We therefore opt for a simple asymmetric decoder composed of minimalistic upsampling modules, which offers enough recognition power to match the accuracy of heavyweight approaches [66] while supporting real-time inference.

Early approaches to enlarge the receptive field of dense predictions were based on dilated convolutions [29, 75, 77]. A more involved approach is known as spatial pyramid pooling (SPP) [78]. SPP averages features over aligned grids with different granularities [79]. Our baseline approach uses a convolutional adaptation of that idea as proposed in PSPNet [15] and elaborated in Ladder-DenseNet [66], where a feature pyramid is upsampled and concatenated with input features. Thus, subsequent convolutions obtain access to broad spatial pools which increase their receptive field. The receptive range can also be enlarged by applying a bank of atrous convolutions with different rates, which is known as à trous SPP, or ASPP for short [14]. SPP has been recently improved by processing each pooled representation with a squeeze-excitation module [80]. Large receptive range can also be obtained by applying the convolutional backbone to different levels of the resolution pyramid [28]. The recovered scale-covariant representations can be fused by a scale selection multiplexer controlled with trained attention [75] or depth information [38]. However, most recent attention has been directed towards simpler fusion schemes based on addition [18, 81]. In this thesis, we propose pyramidal fusion as a novel and principled approach to fuse a pyramidal representation with shared parameters. Pyramidal fusion has several distinct advantages which we summarize towards the end of this section.

There are many approaches to reduce computational load of a convolutional model while preserving its accuracy. Quantization approaches reduce the weight precision [82]. Pruning approaches reduce the number of connections within the model [83]. However, in this thesis we mostly experiment with simplified forms of convolution. Some forms of simplified convolutional forms include grouped convolutions and depthwise separable convolutions. Grouped convolutions reduce the number of floating point operations and the number of parameters by enclosing the information flow within smaller groups of feature maps. Depthwise separable convolutions [84, 85] decrease computational complexity by splitting a regular convolution in two. Firstly, a $k \times k$ convolution is separably applied to each input channel. This can be viewed as a group convolution where the number of groups equals the number of channels C (there are C kernels k×k×1). Secondly, a 1×1 convolution is applied to propagate inter-channel information. Replacing standard convolutions with depthwise separable convolutions lowers the number of parameters and decreases computational complexity at the cost of some performance drop. Strong regularization effect of depthwise separable convolutions can be relaxed with inverted residual blocks [43] which consist of three layers: i) a projection which expands dimensionality to C, ii) a grouped convolution with C groups, iii) a linear projection which restores the initial dimensionality. Our experiments confirm that inverted residual blocks lead to compact and accurate models which are suitable for mobile applications, although they are still less efficient than standard convolutions on popular GPU hardware. Various methods have been proposed to discover prominent inter-group connections. ShuffleNet [71] uses channel shuffling to pass information across convolutional groups. CondenseNet [86] incorporates a training strategy which locates important connections in grouped convolutions and prunes those which are redundant.

3.2 Efficient architectures for semantic segmentation

Many real-time semantic segmentation approaches refrain from backbones designed for competitive ImageNet performance. ENet [87] proposes an architecture with custom bottleneck residual blocks, and a decoder with no lateral connections. ERFNet [17] proposes residual units composed of two 1D convolutions (3×1, 1×3). DG2S [88] improves on ERFNet with i) depthwise separable convolutions, and ii) grouped projections with channel shuffling. ERF-APSPNet [89] combines the ERFNet encoder with ASPP [14]. ESPNet [19] factorizes convolutions into i) a bottleneck projection, and ii) concatenated atrous convolutions with different dilation factors (similar to ASPP). ICNet [18] proposes a resolution pyramid with partially shared parameters and a custom lightweight encoder. They gradually fuse multi-scale representations before restoring the resolution by a decoder without lateral connections. LERNet [90] introduces a custom residual backbone which uses depthwise separable convolutions to maintain low processing requirements. They improve inference speed in video by propagating information across video frames with an attention module based on feature similarity.

Recent research shows that efficient semantic segmentation models can also be based on lightweight ImageNet pre-trained classification architectures. Ladder-DenseNets [33, 66] combine a customized DenseNet encoder, SPP module, and ladder-style upsampling. They deliver over 80% mIoU on Cityscapes test, and allow training on a single consumer-grade GPU due to small memory footprint of suitably checkpointed DenseNet encoders. LinkNet [91] and RefineNet-LW [92] also use an ImageNet-pretrained backbone and ladder-style upsampling, however, they propose convolutional postprocessing at full resolution [91] and thicker upsampling [92]. GUN [81] proposes a two-level pyramid based on the dilated DRN-D-22 encoder [77] with partially shared parameters. They also use a decoder with three lateral connections and a guided nonlinear upsampling unit which improves accuracy at object boundaries. SwiftNetRN-18 [40] (our baseline model) consists of a ResNet-18 encoder, lightweight SPP [66], and lightweight decoder with ladder-style upsampling [66]. SwaftNet [93] extends SwiftNet with squeeze-excite in lateral connections. DFANet [94] applies a shared Xception-based encoder at multiple scales. However, only the finest level of their encoder observes the image, while all other levels operate on a blend of previously extracted features. DF2-Seg2 [95] proposes a neural architecture search algorithm which separately optimizes the decoder and the encoder of an architecture similar to our single-scale model without SPP. Their models would likely profit from increasing the receptive range with pyramidal fusion or SPP.

Table 3.1 presents time complexity analysis of popular classification models. For each architecture, the total number of multiply-additions is shown for each convolutional block. Moreover, the share in total model computation is displayed for each block.

Table 3.1: Analysis of per-block complexity expressed in the number of multiply-adds for different classification architectures.

Model	Stem	EB 1	EB 2	EB 3	EB 4	Total
densenet121	114M (4%)	1102M (40%)	786M (28%)	629M (23%)	102M (3%)	2733M
densenet161	171M (2%)	2470M (33%)	1763M (23%)	2576M (34%)	446M (6%)	7426M
densenet169	114M (3%)	1102M (33%)	786M (24%)	960M (29%)	278M (8%)	3241M
resnet18	114M (6%)	443M (25%)	393M (22%)	392M (22%)	392M (22%)	1735M
resnet34	114M (3%)	664M (18%)	835M (23%)	1275M (36%)	613M (17%)	3501M
resnet50	114M (2%)	645M (16%)	986M (25%)	1401M (35%)	773M (19%)	3921M
mobilenet_v2	53M (17%)	42M (14%)	30M (9%)	102M (34%)	73M (24%)	300M
vgg16	86M (0%)	1767M (11%)	2649M (17%)	4412M (29%)	4411M (29%)	14767M

We now highlight differences between our method and the most related previous work. Our single-scale baseline (SwiftNet-RN18) is similar to Ladder-DenseNets [66] since it also uses ladder-style upsampling and spatial pyramid pooling. However, in this thesis we focus on pyramidal fusion which outperforms SPP in all our experiments, especially when training data is scarce and image resolution is large. Additionally, we exploit simpler backbones (ResNet-18 and MobileNet-v2) and therefore report almost three times faster inference. In comparison with LinkNet [91], we do not consider convolutions at full resolution and keep a lower number of decoder channels. In comparison with RefineNet-LW [92], our design requires 50% less floating-point operations without sacrificing recognition accuracy on road-driving datasets. In comparison with GUN [81], we embrace subsampling instead of avoiding it through dilated convolutions, since our ladder-style decoder delivers cheap and accurate upsampling.

Pyramidal fusion is a principled architectural pattern which collects representations from *all* processing blocks across *all* pyramid levels and then gradually blends them within the ladder-style decoder. When compared with recent pyramidal approaches ICNet [18] and GUN [81], our model shares all backbone parameters in all three levels of the pyramid and uses twelve lateral connections between processing blocks instead of only two or three. In comparison with DFANet, our model applies the same backbone to all images of a resolution pyramid, while they operate on a single resolution input. To conclude, the main advantage of pyramidal fusion is a clear and principled design which promotes scale covariance, and allows for context recognition on $128\times$ subsampled representation which induces a large receptive range.

Chapter 4

Semantic segmentation with pyramidal fusion

Our method starts from the following assumptions. Recognition encoders should be pre-trained on ImageNet in order to benefit from knowledge transfer. Receptive range should be enlarged by a suitable module or architectural pattern. The resolution of encoded features should be restored by a ladder-style decoder in order for the predictions to retain detail. The upsampling path must be simple in order to support real-time inference. Gradient flow should be facilitated throughout the network to ensure efficient learning. This requirement would also favour training from random initialization in an event that ImageNet pre-training is not available or not useful.

4.1 Basic building blocks

Our contribution is conceived around three building blocks which we introduce in the following paragraphs. These building blocks are going to be used in our two architectures which we describe in subsequent subsections.

4.1.1 Convolutional layers for fast inference

Basic two-dimensional convolution at spatial location (i, j) is described by the following equation:

$$\mathbf{I}(i, j) * \mathbf{K}^f = \sum_c \sum_m \sum_n \mathbf{I}(c, i-m, j-n)\mathbf{K}^f(c, m, n) \tag{4.1}$$

where $\mathbf{I}$ is the input tensor of size $C \times H \times W$ and $\mathbf{K}$ is the kernel tensor of size $F \times C \times K \times K$. $\mathbf{K}^f$ is a kernel of size $C \times K \times K$ which produces the f-th feature map in the output tensor. The standard convolution is visualized in Figure 4.1. By convolving all F kernels across the input tensor, the final result is a tensor of size $F \times H \times W$. The regular convolution is consisted of roughly $F \times H \times W \times K^2$ dot products which translates to $F \times H \times W \times K^2 \times C$ fused multiply-accumulate operations(MACs). In the rest of this subsection, we give an overview of operations more efficient in computation or parameter count. We do not take into consideration the complexity of per channel bias additions as their complexity is a few orders of magnitude smaller.

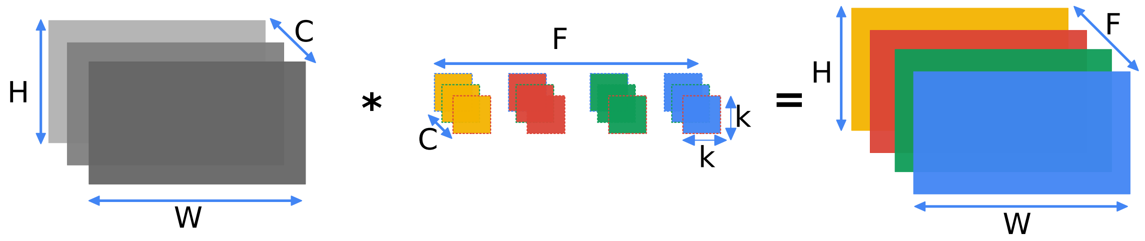

Figure 4.1: Visualization of feature maps produced by a standard convolution. All kernels (small squares) are applied at all spatial locations of the all input feature maps (gray rectangles). Each output feature map is produced by the kernel of the same color.

Group convolution operates on parts of the input tensor rather than the whole tensor, as shown in Figure 4.2. These parts are called groups. The number of convolutional groups is inversely proportional to number of operations. For an input tensor of size $C \times H \times W$, G groups, where each group is convolved with a $\frac{F}{G} \times \frac{C}{G} \times K \times K$ sized kernel, there is $\frac{F}{G} \times H \times W \times \frac{C}{G} \times K^2$ MACs per group. In total, there is $F \times H \times W \times \frac{C}{G} \times K^2$ MACs in a group convolution. The standard convolution may be viewed as a grouped convolution with $G = 1$.

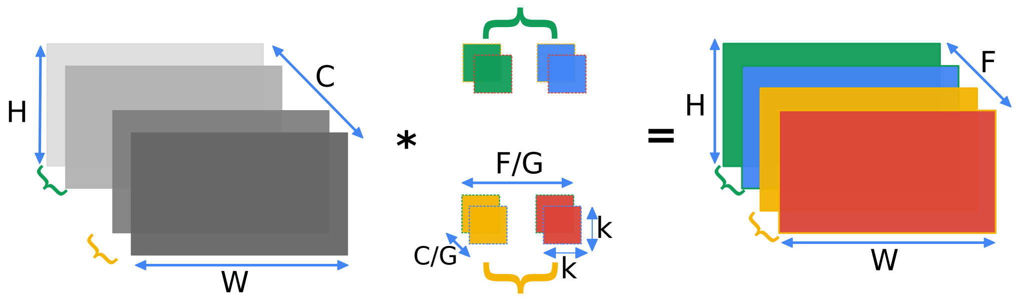

Figure 4.2: Visualization of feature maps produced by a grouped convolution. Each kernel (small squares) is applied at all spatial locations of input feature maps (gray rectangles) in its group (brackets with matching color). Each output feature map is produced by the kernel of the same color. Note that there is no dependency between output feature maps produced by kernels from different groups.

Depthwise separable convolution. On the other extreme, the convolution where the number of input channels equals the number of convolutional groups, i.e. $G = C$, is called a separable convolution. Such convolution does not have any information shared between different convolutional groups which greatly limits model expressivity. By introducing a 1×1

convolution after the preceeding separable convolution, connections between groups are made and inter-group information flow is supported. This 2-convolution sequence is called a depthwise separable convolution and is visualized in Figure 4.3. Separable convolution consists of $F \times H \times W \times K^2$, and the pointwise convolution consists of $F \times H \times W \times C$ MACs. Together, there is $F \times H \times W \times (K^2 + C)$ MACs in a depthwise separable convolution.

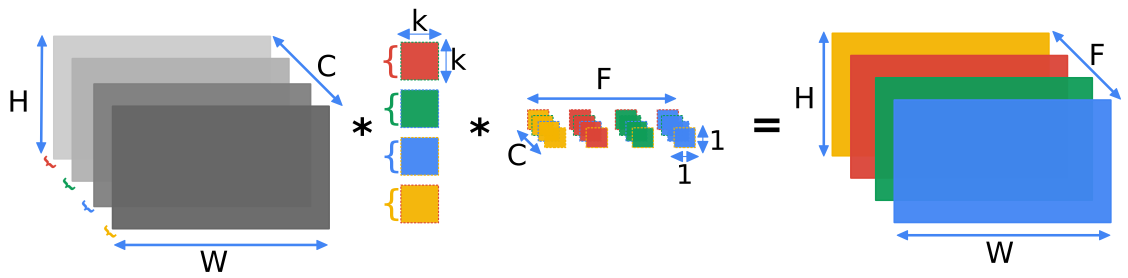

Figure 4.3: Visualization of feature maps produced by a depthwise separable convolution. Each kernel (small squares) is applied at all spatial locations of a single input feature map (gray rectangle). Afterwards, a standard convolution with kernel size 1 is applied to the intermediate representation.

Fusing batch normalization with convolution. Batch normalization (BN) [56] commonly used in deep models since introduced. It is shown that BN alleviates the problem of vanishing gradients and has regularization effects. Batch normalization is applied at each channel separately in the following manner:

$$BN_{\gamma,\beta}(\mathbf{x}_i) \equiv \gamma \frac{\mathbf{x}_i - \mu}{\sqrt{\sigma + \varepsilon}} + \beta \tag{4.2}$$

where $\mathbf{x}_i$ is the input tensor, μ and σ are the mean and variance of feature map activations, and γ and β are trainable parameters which allow BN to learn an identity function. During training, μ and σ are calculated for each mini batch separately, where a moving average of their values is recorded. During inference, these global averages are used to perform batch normalization. The shift and scale operation may be moved to the preceding convolutional layer by modifying the kernel and the bias parameters. This way, there is no need to perform batch normalization separately. When fusing BN to preceding convolutional kernel for inference, there is no additional computational strain introduced during inference. However, during training, BN activations contribute to roughly 50% of the total memory allocated. This is the main bottleneck when training BN supported convolutional models.

4.1.2 Elements of an efficient encoder-decoder architecture

Accurate segmentation architectures are composed of lightweight convolutional instances. Two parts of the architecture are highlighted: an encoder and a decoder. The encoder processes the input image to form a spatially coarse but high-dimensional tensor. High dimensionality is measured by the number of feature maps at the encoder output, which is several orders of magnitude higher than the number of image channels. Spatial resolution is recovered using a decoder. The proposed architecture uses a series of simple upsample-blend operations which are suitable for efficient inference. Finally, adjusting the receptive field while enabling fast inference is necessary to enable segmentation accuracy in large images.

Recognition encoder We consider compact ImageNet-grade architectures which offer fair performance and affordable computational requirements. We focus on ResNet-18 [42] and MobileNet V2 [43] for a number of reasons. They are a good fit for fine tuning due to pre-trained parameters being publicly available. They promote efficient training from scratch due to moderate depth and residual structure. Finally, they are compatible with real-time operation due to relatively low complexity. Computationally, ResNet-18 is around six times more complex, however the two models are equally fast on GPU hardware which is likely due to depthwise separable convolutions being less efficient than regular convolutions. Both encoders consist of four encoder blocks (cf. EB in Fig.4.5) which produce intermediate features on subsampling levels `x4`, `x8`, `x16` and `x32`.

Upsampling decoder The recognition encoder transforms the input image into semantically rich visual features. These features must have a coarse spatial resolution in order to save memory and processing time. The purpose of the decoder is to upsample these features to the input resolution. We advocate a simple decoder organized as a sequence of upsampling modules (cf. UP in Fig.4.5) with lateral connections [33]. The proposed ladder-style upsampling modules have two inputs: i) low-resolution features from the preceding upsampling module, and ii) high-resolution features from the corresponding encoder block. The low-resolution features are first upsampled with bilinear interpolation to the same resolution as the lateral features coming from the encoder. Upsampled input features and lateral encoder features are then fused with elementwise summation and finally blended with a 3×3 convolution. Note that replacing that convolution with either a 1×1 convolution, or a depthwise separable convolution decreases the validation accuracy.

We route lateral features from the last convolutional unit of each encoder block. In the case of ResNet-18, we use features from the last addition operator as shown in Figure 4.4. Note that routing lateral features from the subsequent ReLU decreases the validation accuracy in our multi-scale setup.

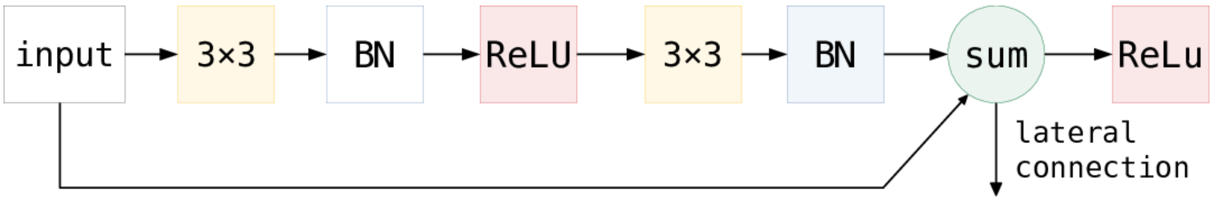

Figure 4.4: Structural diagram of the last residual unit of an encoder block. We do not use pre-activation [96] since we could not find a pre-trained parameterization for ResNet-18. The lateral connection is taken from the output of the elementwise sum after the last residual block. The output of the ReLU node is forwarded to the next encoder block. Credit: Oršić and Šegvić [41].

Increasing the receptive field We consider two approaches for increasing the receptive field while maintaining real-time speed: i) spatial pyramid pooling (SPP), and ii) pyramidal fusion. SPP [15, 66] produces feature maps with varying level of detail by enriching features from the encoder output with their pools over coarse spatial grids 1×1, 2×2, 4×4 and 8×8. Pyramidal fusion is based on genuine multi-scale representations which we train with boundary-aware loss [35, 97] in order to avoid overfitting to unsuitable level of detail. We propose to blend representations at different levels of abstraction and thus enlarge the receptive field without sacrificing spatial resolution.

4.2 Single-scale architecture

Our single-scale architecture transforms the input image into dense semantic predictions throughout downsampling recognition encoder, spatial pyramid pooling module, and lightweight upsampling decoder, as shown in Figure 4.5. Yellow trapezoids designate encoder blocks (EB), that is, parts of the backbone which produce the same spatial resolution on output. All considered encoders consist of four such blocks. The first block produces features at the H/4×W/4 resolution, while each following block increases subsampling by the factor of 2. Thus the features at the far end of the encoder are H/32×W/32. These features are fed into the spatial pyramid pooling module (SPP) (cf. green diamond in Fig. 4.5) in order to increase the effective receptive field. The resulting tensor is finally routed to the decoder whose upsampling modules (UP) are shown in blue.

Our SPP module is a simplified and slightly improved version of the pyramid pooling module (PPM) from PSPNet [15], as proposed in [66]. Most differences between the two modules are caused by their different placements in the two architectures. PSPNet has only one convolutional layer between the PPM and the predictions. Hence, PPM has two responsibilities: to provide context and to condition features for linear classification. On the other hand, in our architecture, SPP features must pass through the entire decoder before getting classified in the lower-left part of Fig. 4.5. Hence, our SPP is just an instrument for enlarging the receptive field. Therefore, we reduce the input dimensionality to 128 (PPM has 2048 feature maps on input),

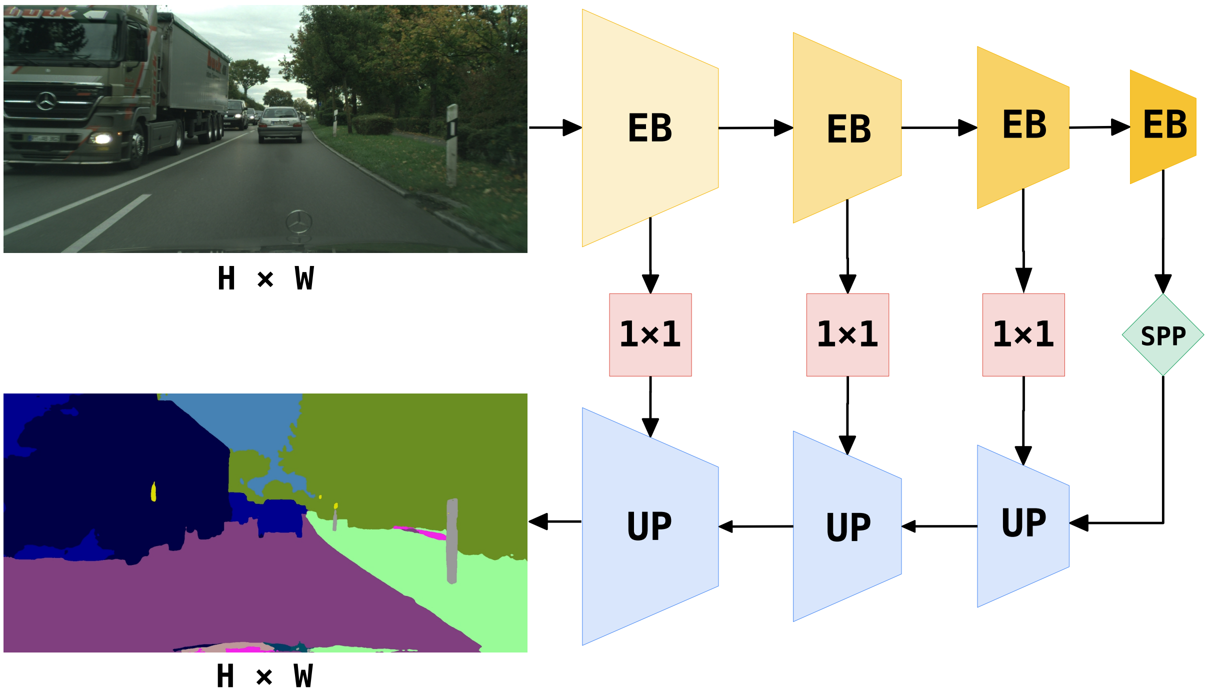

Figure 4.5: Structural diagram of the proposed single scale model. Yellow trapezoids designate encoder blocks which may be pre-trained on ImageNet. Green diamond designates the spatial pyramid pooling module, red squares designate bottleneck projections, while blue trapezoids designate lightweight upsampling modules. Logits are upsampled to the input resolution with 4× bilinear interpolation. Credit: Oršić and Šegvić [41].

and produce 128 feature maps on output (the output dimensionality of PPM is 4096). This reduction is appropriate since our SPP operates on H/32×W/32 input which corresponds to 16 times less data than PPM which operates on H/8×W/8. Besides decreasing the capacity, we also adapt the four subsampling grids to the aspect ratio of the input tensor. Thus our coarsest pool is 1×1 during training on square crops, while its shape is 1×2 during inference on Cityscapes.

Note that our decoder and encoder are asymmetric [33]: the encoder has many 3×3 convolutions per encoder block while the decoder has only one 3×3 convolution per upsampling module. Furthermore, the dimensionality of encoder features increases along the downsampling path, while the dimensionality of the decoder features is constant. As discussed in the related work, this design reflects our belief that recognition requires more capacity than locating borders when the semantic content is known. Therefore, lateral connections have to adjust dimensionality with 1×1 convolutions which we designate with red squares. The decoder uses a single 3×3 convolution per upsampling stage in order to keep inference time as low as possible. Upsampling modules operate in three steps: i) the low-resolution representation is bilinearly upsampled, ii) upsampled representation is summed with the lateral connection, iii) the summation result is blended with 3×3 convolution. This architecture presents a very strong baseline, especially when equipped with an efficient encoder such as ResNet-18 or MobileNet V2 [40].

4.3 Multi-scale architecture with pyramidal fusion

Our multi-scale architecture independently extracts features at different levels of the resolution pyramid as shown in Figure 4.6. All instances of the recognition encoder share parameters in all four processing blocks. This enforces scale covariance within the encoder (unlike SPP), which allows to recognize objects of different sizes with the same set of parameters. This also increases the receptive field since our predictions have access to features extracted from coarse images. We expect that such design will get more and more prominence as image resolutions get larger. Finally, entanglement of heterogeneous representations enhances gradient propagation towards early layers and promotes ensemble-like behaviour as we show in experiments.

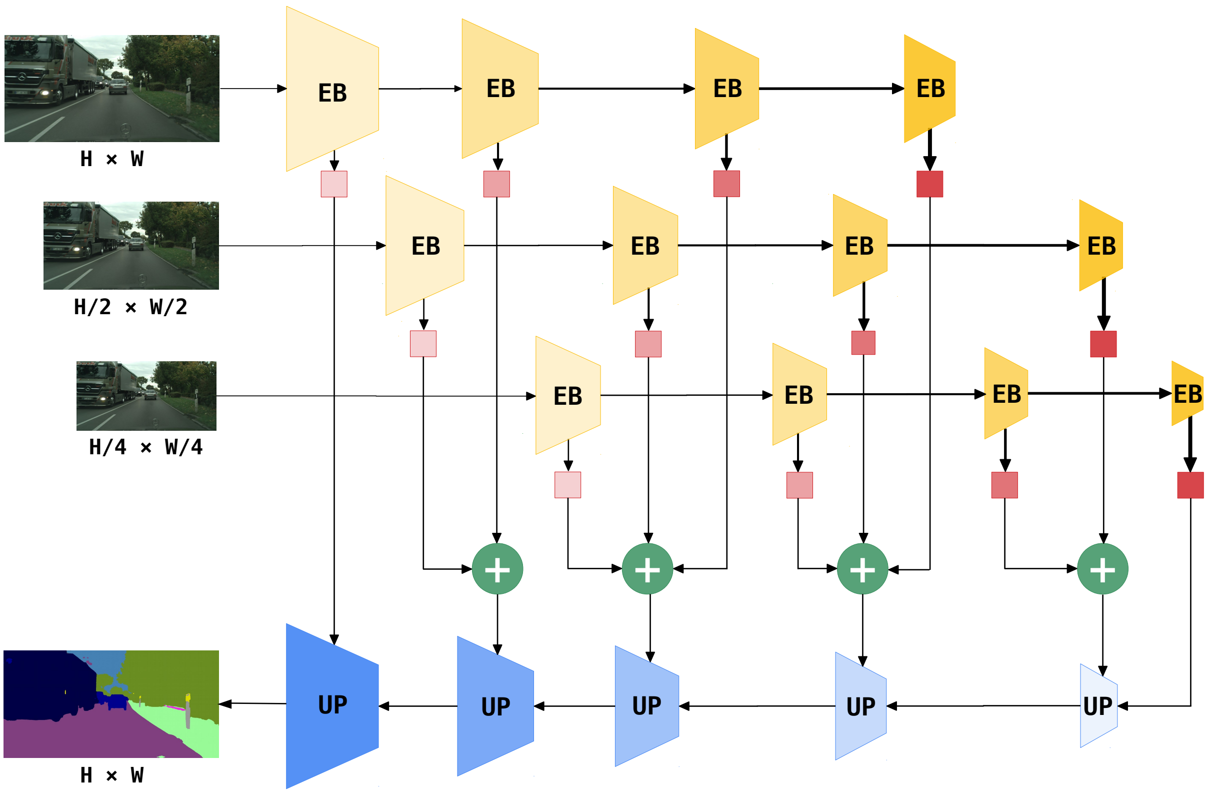

Figure 4.6: The proposed multi-scale architecture has shared encoders and pyramidal fusion. Yellow trapezoids denote encoder blocks (EB). Red squares denote projections (1×1 convolutions) which establish dimensionality of the decoder. Green circles represent elementwise summation. The resolution is restored by lightweight upsampling modules designated with blue trapezoids (UP). Same color indicates shared parameters. Credit: Oršić and Šegvić [41].

Yellow trapezoids denote encoder blocks (EB). Corresponding instances of the same encoder block are designated with the same colour as a hint that they share parameters. Red squares denote projections (1×1 convolutions) which adjust the number of feature maps to the dimensionality of the upsampling path. Connection widths illustrate feature dimensionality. Green circles denote elementwise summation which fuses all extracted features at the same spatial resolution. This step is our main novelty and we denote it as pyramidal fusion. The fused features are passed to the decoder as lateral connections for ladder-style upsampling. As in the single-scale architecture, the upsampling modules involve bilinear upsampling, elementwise addition and 3×3 convolution. The logits are produced by simple projection and 4x bilinear upsampling.

Pyramidal fusion may look queer since it fuses heterogeneous features with different semantics. However, such arrangement results in a very well connected model which trains well and favours generalization due to acting as a large ensemble of simpler models. Early experiments with this architecture underperformed on small objects [40], which we attribute to overfitting to the coarsest level of the resolution pyramid. We counteract this effect by favouring pixels which are close to semantic boundary, as we show next.

4.4 Increasing the penalty for boundary pixels

Adding levels to our image pyramid enlarges the receptive range and regularizes the encoder. The coarsest feature maps created by the 3-level image pyramid are only 8×16 pixels wide. These features contribute by representing a wide context of dense predictions, similar to SPP which also assembles extremely coarse representations (1×1, 2×2, 4×4, and 8×8). However, this may cause two problems for our pyramidal fusion architecture: i) downsampled representations may hurt detection of small objects, ii) increased number of randomly initialized parameters may lead to overfitting. We attempt to address these issues by guiding the optimization procedure to prioritize pixels at semantic borders.

We emphasize the importance of pixels near semantic boundaries by employing the boundary-aware loss [35]. This is a modification of the focal loss [98] which amplifies the loss at pixel (i, j) according to the distance d^{ij} from that pixel to the closest semantic border. We determine d^{ij} by applying distance transform to the ground truth segmentation labels. The modulation is expressed by the boundary factor α^{ij} which is formulated as follows:

$$\alpha^{ij} = \begin{cases} 8, & \text{if } d^{ij} \in [0, 15] \\ 4, & \text{if } d^{ij} \in [16, 63] \\ 2, & \text{if } d^{ij} \in [64, 127] \\ 1, & \text{if } d^{ij} > 127 \end{cases} \tag{4.3}$$

The boundary-aware loss multiplies the standard negative log likelihood of the correct prediction with two pixel-level modulation factors:

$$L^{ij} = -\alpha^{ij} \mathrm{e}^{\gamma(1-p_t^{ij})} \log p_t^{ij} \ . \tag{4.4}$$

Here, p_t^{ij} is the probability the model assigned to the ground truth class. Modulation factor of the focal loss $\mathrm{e}^{\gamma(1-p_t^{ij})}$ gives precedence to poorly classified pixels (we use γ=0.5 in all ex-

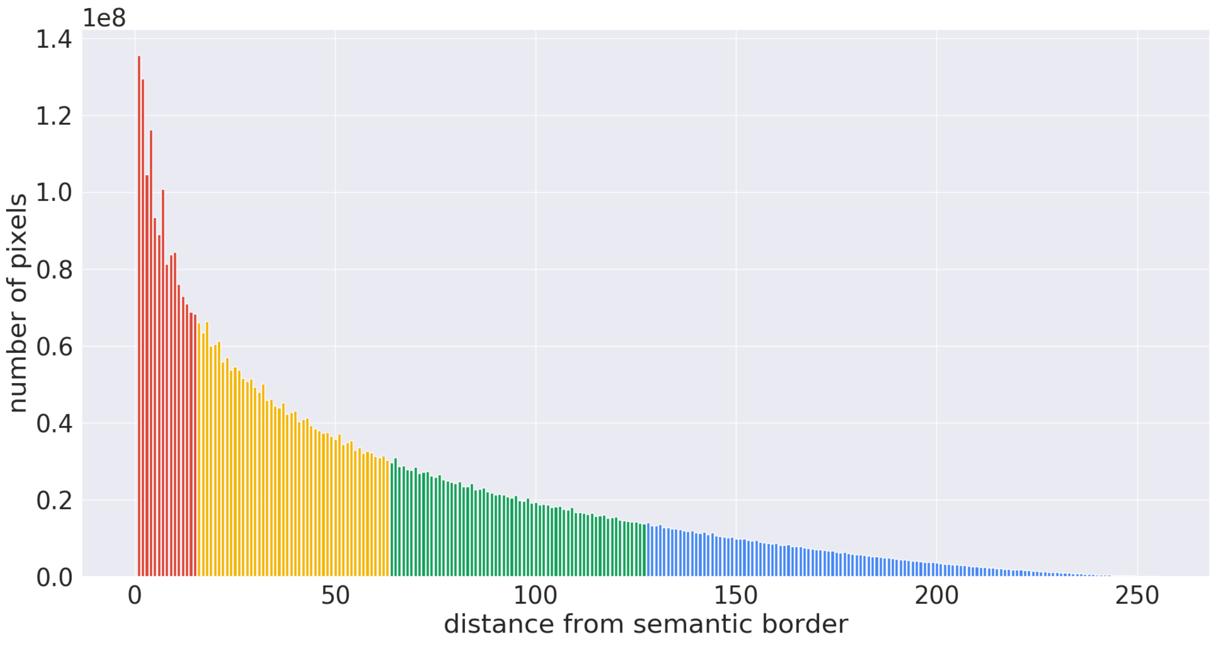

Figure 4.7: Histogram of distances to the closest semantic border on Cityscapes train at full resolution. Plot colors designate the four loss weights resulting from thresholds from (4.3). Credit: Oršić and Šegvić [41].

periments). Finally, the boundary factor α^{ij} prioritizes correct predictions at semantic borders. Figure 4.7 shows a histogram of distances from semantic borders (d_{ij}) across full resolution Cityscapes train. A large proportion of Cityscapes pixels are close to a semantic border. The binning procedure (4.3) leads to almost equal frequency of the four weights. This is demonstrated in Figure 4.8. The figure visualizes maps of the boundary factor α obtained with bins from (4.3). Note how color coding matches the histogram colors from Figure 4.7.

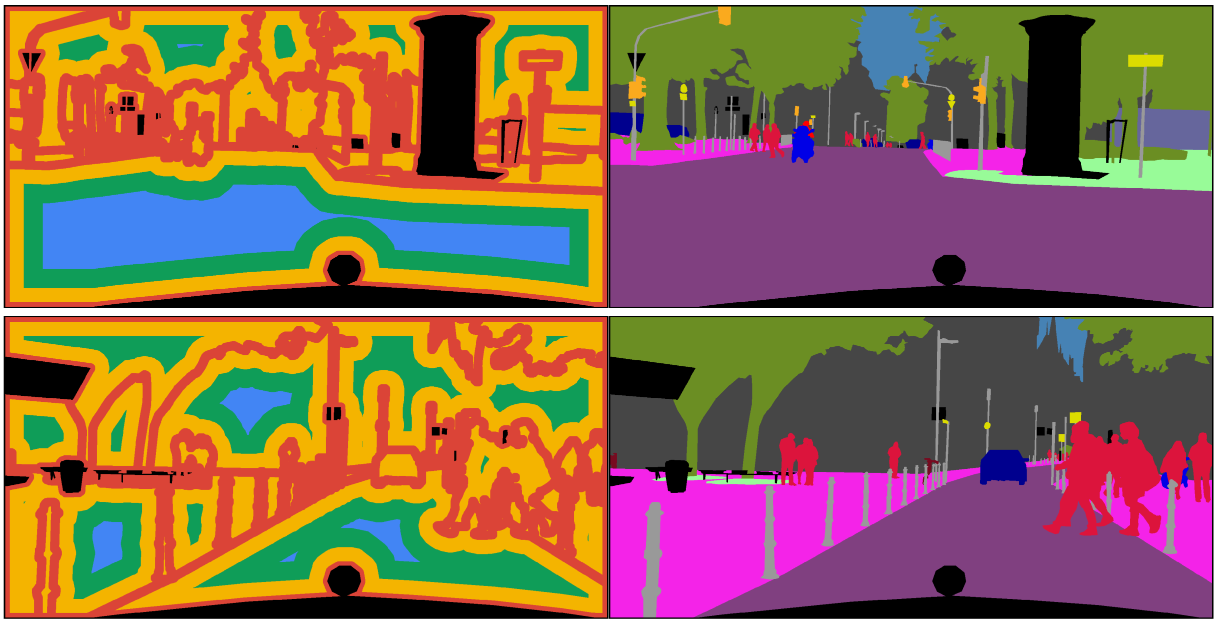

Figure 4.8: Visualization of the boundary factor α (left) as determined from ground truth labels (right) according to (4.3). The loss weights are designated with the same colours as in Figure 4.7. The closer a semantic boundary — the greater the loss multiplier. Credit: Oršić and Šegvić [41].

Chapter 5

Experiments

We evaluate mIoU accuracy of our methods in experiments on four semantic segmentation datasets: Cityscapes [23], CamVid [99], Mapillary Vistas [24] and ADE20k [100]. We report the inference speed of our models on two desktop GPUs (GTX 1080Ti and RTX2080Ti) and two embedded GPUs (Jetson TX2 and Jetson Nano), as well as their computational complexity as determined by `torchstat`[*]. We present ablation and validation experiments which provide a more detailed insight into the impact of various design choices. Finally, we present experiments which interpret the decision procedure of our models on the Cityscapes dataset.

5.1 Training and inference details

We train our single-scale models with the usual cross-entropy loss and add the boundary-aware term for models with pyramidal fusion. Single-scale models are unaffected by the boundary-aware loss. All pyramid models share encoder parameters. However, batch normalization statistics are independently calculated at different levels of the pyramid. We use the Adam [101] optimizer. We set the initial learning rate to $4 \cdot 10^{-4}$ and decay it with cosine annealing to the minimum value of $1 \cdot 10^{-6}$ in the last epoch (we do not perform any warm restarts). The weight decay is set to $1 \cdot 10^{-4}$. In experiments with ImageNet pre-training, we update pre-trained parameters with a 4 times smaller learning rate and apply 4 times smaller weight decay. We train on jittered square crops with batch size 14. Jittering consists of random horizontal flipping, and scaling with random factors between 0.5 and 2. We set the random crop size to 768×768 for full Cityscapes resolution and Vistas, 448×448 for half Cityscapes resolution and CamVid, and 384×384 for ADE20k. On Cityscapes, Vistas and ADE20k, we employ a random crop sampling strategy which favors pixels of rare classes [102]. We train for 250 epochs on Cityscapes and 400 epochs on CamVid. When training from scratch, we train for additional 200 epochs. We train Vistas and ADE20k models for 100 epochs.

[*] `https://github.com/Swall0w/torchstat`

We convert our binary PyTorch models to ONNX format and subsequently optimize them with TensorRT. Effects of TensorRT optimization vary across models and platforms. A TensorRT version of ResNet-18 is only slightly faster than the corresponding PyTorch version with fused batch normalization layers [40]. In case of MobileNet V2 and embedded hardware, the effects are significantly larger. Under PyTorch, we simulate real-time application by setting batch size to 1. We measure the time elapsed between transferring the input data to the GPU and receiving the semantic predictions into RAM as shown in Figure 5.1. Under TensorRT, we measure the inference speed with the equivalent C++ code.

```
device = torch.device('cuda')
model.eval()
model.to(device)
with torch.no_grad():
input = model.prepare_data(batch).to(device)
logits = model.forward(input)
torch.cuda.synchronize()
t0 = perf_counter()
for _ in range(n):
input = model.prepare_data(batch).to(device)
logits = model.forward(input)
_, pred = logits.max(1)
out = pred.data.byte().cpu()
torch.cuda.synchronize()
t1 = perf_counter()
fps = n / (t1 - t0)
```

Figure 5.1: The proposed procedure for measuring the inference speed under PyTorch.

5.2 Cityscapes

The Cityscapes dataset is a collection of images taken during daytime and fine weather from the driver's perspective. It consists of 2975 training, 500 validation, and 1525 test images. Each image has 1024×2048 pixels, and each pixel is assigned one of 19 class labels. Cityscapes also includes 20000 coarsely labeled images which we do not use in our experiments.

Table 5.1 evaluates the accuracy (class mIoU) and efficiency (GFLOP, fps) of our methods when training on full resolution Cityscapes train. Our single scale method based on ResNet-18 achieves 75.4% val mIoU, and delivers 41.0 fps. The corresponding pyramidal fusion model achieves 76.4% validation mIoU and runs at 34.0 fps. All inference speed metrics assume GTX 1080 Ti unless stated differently. The single scale and pyramidal fusion models based on MobileNet V2 encoder reduce the number of parameters and floating point operations while achieving competitive accuracy.

Table 5.1: Semantic segmentation performance on full resolution images from Cityscapes val. Column fps shows the inference speed (frames per second) on GTX 1080 Ti. Column GFLOP denotes the number of floating point operations.

backbone	method	mIoU	fps	GFLOP	parameters
ResNet-18	pyramid	76.4	34.0	128	12.0M
MobileNet V2	pyramid	77.4	29.7	42	2.7M
ResNet-18	single scale	75.4	41.0	114	11.8M
MobileNet V2	single scale	75.3	39.4	39	2.6M

Table 5.2 compares our models with efficient approaches from the literature. Column res denotes resolution at test time. Column fps* provides a rough estimate on how would other methods perform on our hardware. The scaling factors are: 1.0 for GTX1080Ti, 0.61 for TitanX Maxwell, 1.03 for TitanX Pascal, and 1.12 for Titan Xp. We estimated these factors from public benchmarks available at: `goo.gl/N6ukTz`, `goo.gl/BaopYQ`. The column GFLOP* shows an estimated number of floating point operations for an input image of 1MPx, as a resolution-agnostic metric of computational complexity. The two sections of the table presents models which train from scratch (top), and models pre-trained on ImageNet (bottom). For orientation purposes, the bottom section includes two heavyweight approaches — PSPNet [100] and DeepLabv3+ [14], as well as one near-real-time approach — LDN-121 [66].

In both sections of the table, our pyramidal fusion models achieve the best accuracy among methods which deliver more than 30 fps on a GTX1080 Ti on full Cityscapes resolution. Our best model SwiftNetRN-18 pyr achieves 75.9%mIoU and 76.4%mIoU on the Cityscapes evaluation server. Note that the latter result was achieved after pre-training our model on Vistas.

A comparison of our models in different sections of the table reveals that ImageNet pre-training brings at least 5 mIoU percentage points (pp). This indicates that pre-training is an important ingredient on small datasets such as Cityscapes. However, we note that some other designs get less benefit from ImageNet; for instance, ERFNet [17] gets only 1.7% pp. We believe that ERFNet's incapability to benefit from ImageNet pretraining is likely due to insufficient receptive range (ERFNet does not have SPP) and lack of skip connections. The former degrades recognition of smooth surfaces on large objects, while the latter disables recognition of small objects regardless of recognition power. The greatest differences between ERFNet and SwiftNet-RN18 occur at very large objects (e.g. truck: 17 pp mIoU) and small objects (e.g. traffic-light: 8pp mIoU).

Table 5.2: Semantic segmentation performance on Cityscapes when initializing from scratch (top), and from parameters pre-trained on ImageNet. We report input resolution (res), evaluation split (set), achieved accuracy (mIoU), inference speed (fps), normalized inference speed (fps*), normalized computational complexity (GFLOP*) and number of parameters (params). Label pyr denotes the pyramidal fusion model presented in 4.3. Label ens denotes the ensemble of the single scale model and the pyramid model. Symbols † and ‡ designate pre-training on ImageNet and Vistas, respectively. Methods denoted with ⋆ do not publish computational complexity so there we only report GFLOPS and parameters for the encoder.

Model	res	set	mIoU	fps	fps*	GFLOP*	param
ENET [87]	full	test	58.3	21.6	35.4	17.4	1.4M
ERFNet [17]	half	test	68.0	11.2	18.4	55.4	20.0M
D* [88]	half	val	68.4	-	-	11.6	0.5M
DG2s [88]	half	val	70.6	-	-	38.0	1.2M
ESPNet [19]	half	test	60.3	112	108.7	-	0.4M
ICNet [18]	full	test	69.5	30.3	49.7	-	-
LERNet [90]	half	test	66.5	100	100	25.4	0.65
SwiftNetRN-18	half	val	65.3	134.9	134.9	52.0	11.8M
SwiftNetRN-18	full	val	70.4	39.9	39.3	52.0	11.8M
SwiftNetRN-18 pyr	full	val	**72.2**	34.0	34.0	64.0	12.0M
DeepLab v3+(X-65)†⋆ [14]	full	val	79.1	-	-	708.0	38.0M
PSPNet(RN101)†⋆ [15]	full	test	78.4	-	-	722.0	45.0M
LDN-121† [66]	full	test	79.3	15.0	14.5	75.4	9.0M
ERFNet† [17]	half	test	69.7	11.2	18.4	55.4	20M
GUN† [81]	half	test	70.4	37.3	33.3	-	-
DFANet A† [103]	1MPx	val	71.9	100	89.3	1.7	7.8M
DF2-Seg2† [95]	full	test	75.3	32.2	56.3	-	-
SwiftNetRN-18†	half	val	70.2	134.9	134.9	52.0	11.8M
SwiftNetRN-18†	full	test	75.5	41.0	41.0	52.0	11.8M
SwiftNetRN-18 pyr†	full	test	**75.9**	34.0	34.0	64.0	12.0M
SwiftNetRN-18 pyr†‡	full	test	**76.4**	34.0	34.0	64.0	12.0M
SwiftNetRN-18 ens†	full	test	76.5	18.4	18.4	116.0	24.7M

5.3 CamVid

The CamVid dataset contains 701 densely annotated frames. We use the usual split into 367 train, 101 validation and 233 test images. We train on combined train and validation subsets and evaluate semantic segmentation into 11 classes on the test subset. Table 5.3 shows that we obtain an improvement of 1.1 and 1.8 pp mIoU when using the pyramid model with pre-trained ResNet-18 and MobileNet V2 backbones.

Table 5.3 further indicates that ImageNet pre-training contributes more on CamVid than on Cityscapes (7-9pp of mIoU performance). This is not surprising since CamVid has almost 20 times less training pixels. A small size of the dataset poses a considerable challenge when training from scratch due to high overfitting risk. Note that pyramidal fusion brings larger improvement when training from scratch. This suggests that sharing encoder parameters across pyramid levels especially pays off when the training data is scarce.

Table 5.3: Semantic segmentation accuracy on CamVid test. Columns mIoU† and mIoU show the accuracies achieved with ImageNet pretraining and random initialization, respectively.

backbone	model	mIoU†	mIoU
ResNet-18	single scale	72.6	63.3
	pyramid	**73.7**	65.7
MobileNet V2	single scale	71.6	64.0
	pyramid	73.4	65.0

5.4 Mapillary Vistas

Mapillary Vistas [24] is a large road-driving dataset which poses a challenge for real-time methods due to their small capacity. The dataset features 66 semantic classes in 18 thousand training and 2 thousand validation images. To the best of our knowledge, we are the first to report results with a method capable of real-time operation.

We follow the training procedure from subsection 5.1 with the following modifications. During training, we scale the longer side of the input image to 1920 pixels. We collect a distribution of distances from semantic labels in all pixels from scaled training images. We recalibrate the boundary-aware loss by setting α from (4.3) so that roughly the same number of pixels get contained in each bin.

The results are presented in Table 5.4. We adopt public PyTorch implementations for ENet [87] and ERFNet [17] and train them using our code base. These results are shown in the first section. The second section presents our single-scale model and our pyramidal fusion model. The last section displays results of heavy-weight methods which are not capable of real-time operation. We see that pyramidal fusion brings larger benefits than in Cityscapes experiments. A closer look at per-class performance reveals that the most frequent classes such as sky, vegetation, building and car get below-average improvements. This suggests that pyramidal fusion is appropriate for poorly balanced datasets.

Table 5.4: Experimental evaluation on the validation subset of Mapillary Vistas.

method	real-time	mIoU
ENet [87]	✓	23.2
ERFNet [17]	✓	28.8
SwiftNet RN-18 single scale	✓	42.4
SwiftNet RN-18 pyramid	✓	**44.8**
InPlace ABN WRN-38 [102]		53.1
GANet RN-101 [97]		**54.2**
Panoptic-DeepLab [104]		**56.8**

5.5 ADE20k

ADE20k [100] is a very large and diverse semantic segmentation dataset. Its scene parsing benchmark contains 150 stuff and object classes with 20210 training and 2 thousand validation images. Images in ADE20k are much smaller than images from Cityscapes and Vistas. The median image size in ADE20k is 512×480 pixels. Therefore, we set the random crop size during data augmentation to 384 pixels.

Table 5.5 presents our experimental results. We see that pyramidal fusion outperforms the baseline, although the benefits are smaller than in Cityscapes, CamVid and Vistas. This suggests that pyramidal fusion is especially suitable for datasets with very large images, where scale covariance and large receptive range make a larger difference.

5.6 Single-scale model execution profile

To obtain a better insight into the execution time of our models, we report separate processing times and GFLOP metrics for the encoder, and the decoder of our single-scale model. Table 5.6 shows results for input resolution of 1024×2048. Note that these measurements have been obtained in pure PyTorch, without TensorRT optimization, which explains the discrepancy with

Table 5.5: Results on the validation subset of ADE20k.

method	real-time	mIoU
SwiftNet RN-18 single scale	✓	34.3
SwiftNet RN-18 pyramid	✓	34.7
SwiftNet RN-18 pyramid boundary-aware	✓	35.0
RefineNet RN-101 [105]		40.2
PDNs-24NB-2Modules RN-101 [106]		41.9
ACNet RN-50 [107]		43.0
Model A2, 2 conv. [108]		43.7
CDN (+S_C+Aug+MS_Flip) [109]		44.0
CCNet RN-101 [110]		45.2

respect to Table 5.1. The table shows that our decoder is twice as fast as the ResNet-18 encoder. We also note a striking discrepancy of time and GFLOPs for the two downsampling paths. ResNet-18 is almost twice as fast than MobileNet V2 despite requiring 6 times more multiplications.

This phenomenon may be related to the fact that the memory required for caching activations during backprop for the MobileNet V2 encoder is 2.4$\times$ larger than for the ResNet-18 encoder. Additionally, there are many large individual tensors in MobileNet V2 due to expansion in inverted residual convolutional units. For instance, the second MobileNet V2 expansion (`conv2_2/expand`) operates on a 1024/2$\times$2048/2$\times$96 tensor during our inference, which requires 192MB. Hence, poor MobileNet V2 performance on GPU hardware may be caused by high memory requirements and, perhaps, early stage of support for depthwise separable convolutions in cuDNN.

Table 5.6: Inference speed along the downsampling (encoder) and the upsampling (decoder) paths. The columns *dn time* and *up time* display the execution times, while the columns *dn GFLOP* and *up GFLOP* show the number of floating point operations for 2MPx images. Runtime measurements are made under PyTorch (no TensorRT optimization) on a GTX1080Ti.

backbone	model	dn time	up time	dn GFLOP	up GFLOP
ResNet-18	SPP	16.1ms	7.9ms	76.1	30.9
	pyramid	23.7ms	7.9ms	97.0	31.0
MobileNet V2	SPP	26.4ms	7.7ms	12.1	26.9
	pyramid	35.0ms	7.6ms	15.8	26.2

5.7 Runtime efficiency on Jetson TX2 and Jetson Nano

Fig. 5.2 reports inference speed (fps) of TensorRT-optimized models on Jetson Nano, Jetson TX2 and RTX 2080 Ti. The table considers two backbones, as well as 32-bit (fp32) and 16-bit (fp16) floating-point precision.

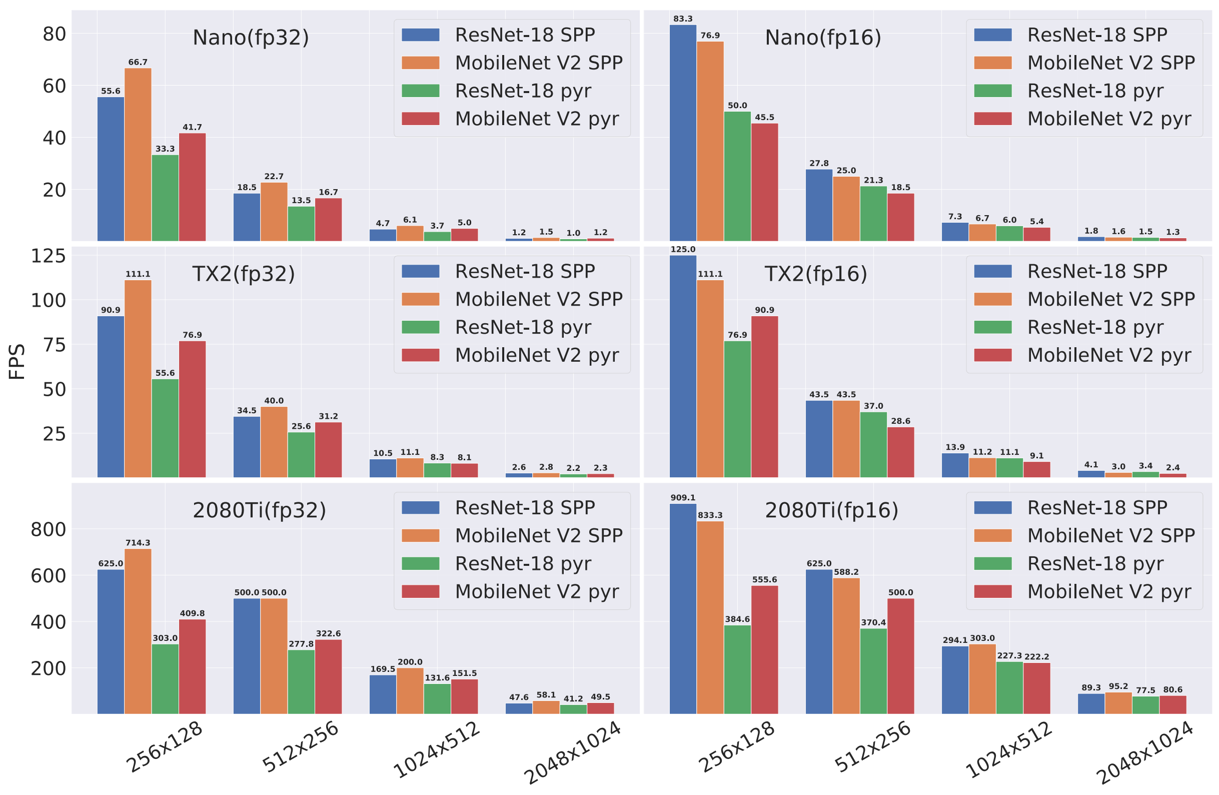

Figure 5.2: Inference speed on Jetson Nano, Jetson TX2 and RTX2080Ti (fps) for two architectures, two backbones, and various input resolutions. All models are optimized with TensorRT under 32-bit (fp32, left) and 16-bit (fp16, right) floating-point precision. Credit: Oršić and Šegvić [41].

We did not measure any validation accuracy drop when performing inference in half precision arithmetic. Both single scale and pyramidal fusion models support real-time inference at 256×512 resolution on both embedded devices. Interestingly, Jetson Nano is only slightly slower than Jetson TX2, although its declared processing power is less than a half than TX2 (0.47 vs 1.3 TFLOP, 25.6 vs 59.7 GB/s). Running in half precision further reduces inference time, especially on Turing microarchitecture (RTX 2080 Ti) where the inference speed is doubled.

5.8 Case study: RoMb Technologies

Experiments from Section 5.7 indicate that the proposed models present clear potential for real-time application on embedded devices. At the same time, our models are suitable for training on small quantities of data due to reasonable capacity. Both of these advantages were tested on the

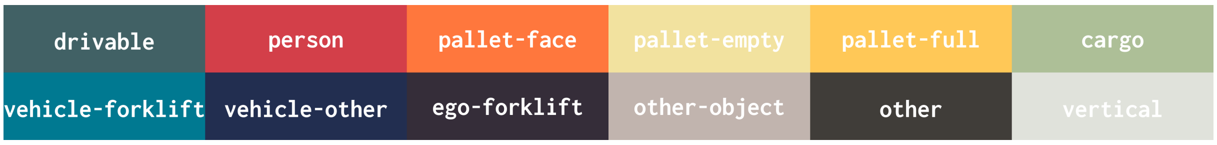

Figure 5.3: Colormap used for visualization of classes in RoMb dataset.

in-house dataset acquired by RoMb Technologies. RoMb Technologies is a Zagreb-based company which develops software for navigating autonomous vehicles in warehouse environments. Their primary goal is to enable reliable navigation of forklifts. Furthermore, they are also interested in visual recognition of loaded and unloaded pallets in order to support their engaging and lifting. Finding objects-of-interest and positioning the forklift for successful operation may be done by annotating driving lanes and pieces of cargo with suitable artificial markers [†]. However, such approach would not be scalable as it would require large human effort. Conversely, it would be much more interesting to allow fully autonomous operation by relying on semantic segmentation of RGB images combined with ultrasonic or laser technologies.

We describe qualitative experiments on an in-house-collected dataset of warehouse scenes. This dataset consists of images acquired using action cameras mounted to a forklift. Depicted scenes contain objects commonly found in warehouses. The dataset is annotated by human operators on the pixel level with polygonal approximations. There are 12 semantic classes which are visualized in Figure 5.3. The main focus of the dataset is to allow training of dense prediction models capable of recognizing pallets and identifying their orientation and state. In particular, the robot should be able to infer whether a given pallet side can be engaged by the forklift as well as whether the pallet is full or empty. Therefore, the dataset taxonomy includes three pallet classes (full, empty and face). The remaining classes in the dataset taxonomy include other vehicles, drivable regions, people, cargo and vertical areas. Figure 5.4 visualizes the distribution of semantic classes throughout the dataset. As in most segmentation datasets, majority of pixels belongs to only a few classes. The most represented classes are *drivable*, *vertical* and *cargo*. The remaining semantic classes are similarly distributed, while only *other* class is underrepresented.

Qualitative results are presented in Figure 5.5. Overall, the system succeeds to deliver useful segmentation maps. However, some cases show the need for more training data. This is most evident in the top row image where the model failed to recognize a partially visible forklift operator. Faulty segmented *ego-forks* in first two examples indicate that changing the camera position and forklift model may effect the segmentation quality. Moreover, in some partially visible pallets (bottom two rows), the segmentation is not complete.

[†] `https://github.com/artoolkit/ARToolKit5`

5.9 Case Study: SafeTram

SafeTram was a project financed by the European Regional Development Fund and featured a partnership between Končar – Electrical Engineering Institute and UniZG FER. Project goals were aimed at developing a system for increasing traffic safety in urban traffic environments

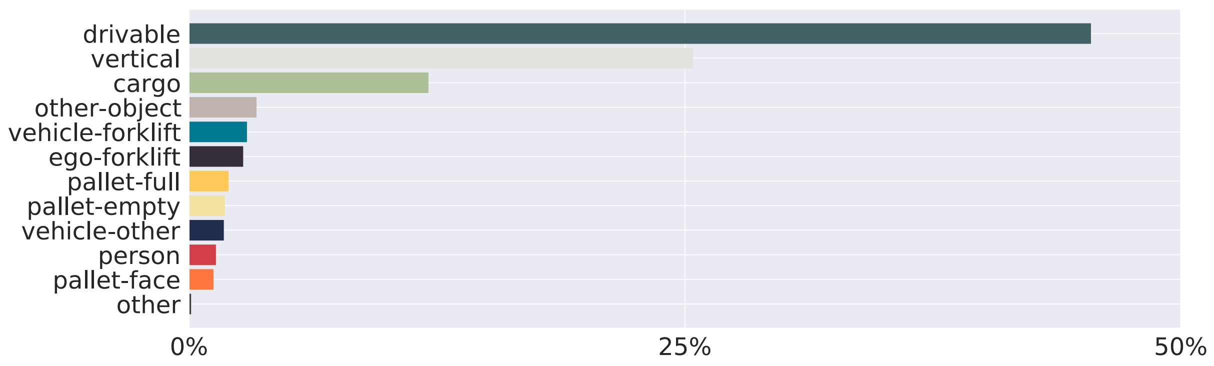

Figure 5.4: Relative distribution of semantic classes throughout the RoMb dataset.

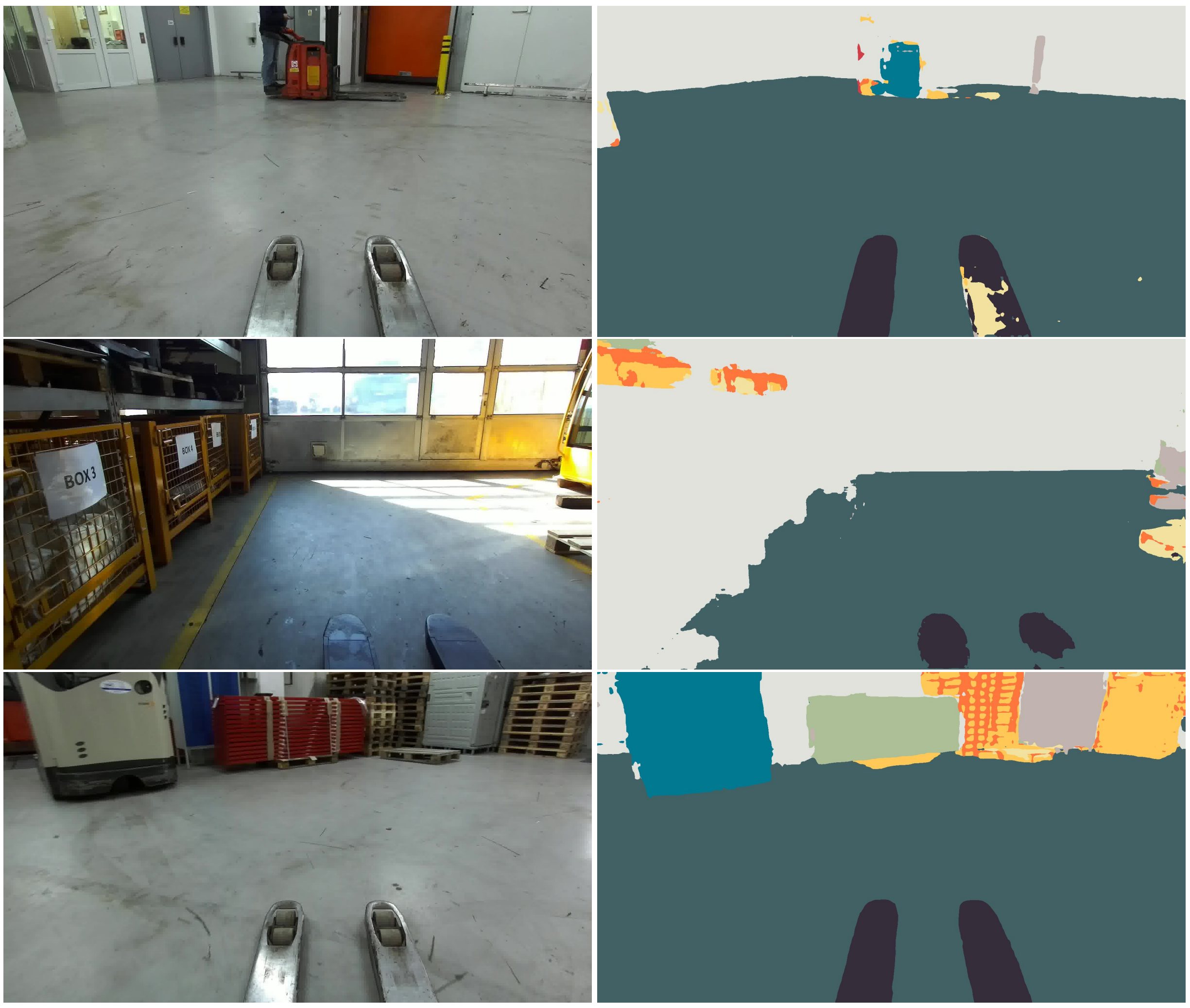

Figure 5.5: Examples of input images (left) and corresponding semantic segmentation on the RoMb dataset (right).

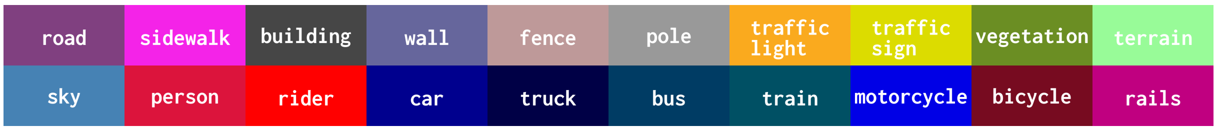

Figure 5.6: Colormap used for visualization of classes in SafeTram dataset.

by equipping electric trams with advanced safety features. One of the project goals included research on computer vision based recognition systems which run in real time when deployed on hardware placed inside trams.

For this and many more purposes, a dataset was collected on urban rail infrastructure in the city of Zagreb. The dataset consists of video sequences in diverse weather conditions. Moreover, video acquisition was carried out during both day and night. Video frame rate was set to 10 FPS, and there are both 8 and 16 bit per pixel camera settings present during recording. In total, there are 116 video sequences which contain 120 133 frames.

An imaging sensor captures red, green and blue components of incoming light by using color filters for each component separately. This may be done for each image pixel on independent sensor planes. Usually this is not the case in modern imaging sensors as it would complicate the manufacturing process. Instead, a single sensor plane filters light captured in each pixel in a pattern best known as a Bayer filter [111]. The pattern consists of 50% green, 25% red and 25% blue pixels. Raw image data from sensor is stored to grayscale (1 channel per pixel) PNGs. A demosaicing procedure should be made in order to read RGB values. Each pixel contains the true value of a single channel whereas the other two values need to be interpolated from neighboring pixels. There are more than one possible Bayer patterns and different interpolation methods which are available in modern computer vision libraries[‡].

A visual recognition system operating in railroad environments should include a "rail-track" class at its output. Large-scale datasets like Mapillary Vistas [24] do not include a sufficient amount of images containing rail tracks. However, Vistas is useful for training due to its diversity and magnitude. The segmentation model is therefore trained concurrently on Vistas and RailSem [112]. RailSem is a publicly available dataset containing images shot from moving trams and trains. As no semantic segmentation labels were collected during creation of the SafeTram dataset, the author had the freedom of creating a label set suitable for training on other publicly available datasets. This label set is visualized in Figure 5.6, and the model was trained concurrently on Vistas and RailSem [112]. RailSem is a dataset most suitable for SafeTram sequences as it contains images from driving railroad vehicles. The label set is defined in two simple steps. First, four RailSem classes corresponding to parts of rail tracks are merged in a single `rails` class. Second, other Vistas and RailSem classes are mapped to one of 19 Cityscapes [23] classes, where possible.

[‡] `https://docs.opencv.org/master/d8/d01/group__imgproc__color__conversions.html`

To achieve the most accurate segmentation accuracy, the single scale model is trained using a large coder, i.e DenseNet-161. Hyperparameters match the Vistas training setup (section 5.4). For memory efficiency, gradient checkpointing of dense layers is used during backpropagation [66].

The following examples study the sensitivity of segmentation quality to different weather and daylight exposure. Figure 5.7 shows the same scenes observed during day and night. The examples show degradation of segmentation quality in trains and underexposed areas which are abundant in night scenery. Moreover, due to slow shutter speed, some scene parts are affected by motion blur and therefore poorly segmented. For example, this effect occurs in moving objects or in the whole scene during camera motion. This is especially evident in scene parts close to the camera, which poses serious limitations to deployment of image based recognition in nighttime conditions.

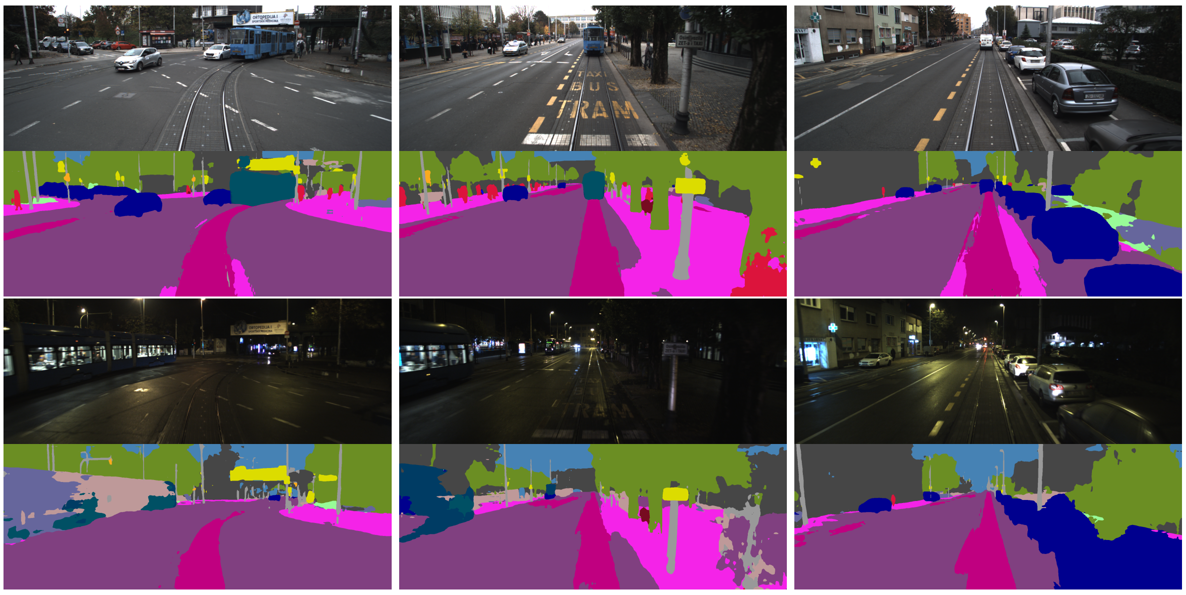

Figure 5.7: Qualitative comparison of model performance in day and night time. Model outputs are visualized for presented input images. Each column compares segmentations of the same road part during day and night. First column displays how night conditions degrade segmentation quality of the *train* class. The middle column shows how pitch black parts of the image make the model clueless. The last column suggests how the model is capable of detecting small pedestrians even at night.

Similarly, Figure 5.8 compares segmentation outputs in corresponding locations during bright, sunny and overcast weather. These examples show how overexposure induced by high dynamic range in the scene affects the segmentation performance. Furthermore, sun flares, which occur when direct light hits the camera lens, may severely impact the segmentation. This is most evident in the example from the last column, where the same bus has different segmentations at the model output.

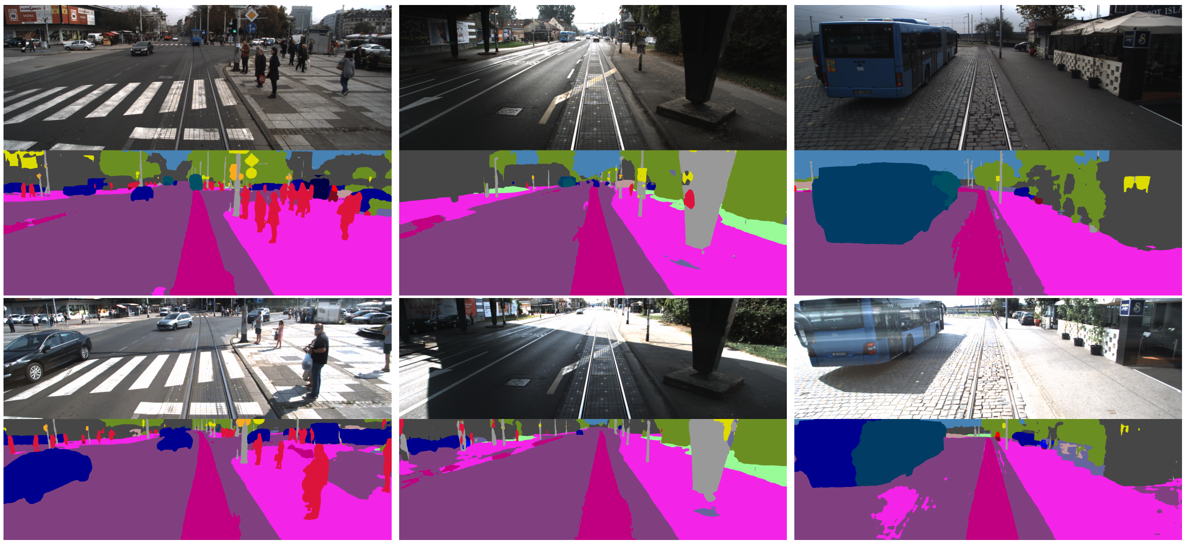

Figure 5.8: Qualitative comparison of model performance during sunny and overcast weather. Model outputs are visualized for presented input images. Each column compares segmentations in different weather conditions. The first column shows how segmentation accuracy degrades in overexposed areas such as the right hand sidewalk, but does not degrade in the middle column example. The rightmost column suggests how flaring in the windshield affects proper segmentation in pixels of the bus.

5.10 Comparison with PPM and ASPP

We study effects of using higher capacity pyramid pooling modules in our single-scale architecture. We train two additional ResNet-18 based models by replacing our SPP with ASPP [14] and PPM [15]. We lower the number of encoder features to 256 feature maps, configure PPM to receive 256 maps, and decrease its output to 128 maps for fair comparison with our SPP module. We set ASPP dilation factors to (3, 6, 9) instead of (6, 12, 18) as in the original work, since our features are 32× subsampled while DeepLabV3+ uses 16× subsampling. Table 5.7 displays inference speed, parameter count and accuracy. We observe that PPM achieves slightly better accuracy but sacrifices inference speed. ASPP, however, degrades accuracy and inference speed. The recognition accuracy is impaired for the following reason. Pyramid pooling in our setup occurs at smaller resolution than in DeepLabv3+. Hence, training many parameters with relatively few data points poses an overfitting risk.

Table 5.7: Comparison of pyramid pooling variants for our single-scale model on Cityscapes val. We report the number of parameters, inference speed at 2MPx and mIoU accuracy.

pooling module	#params	total time (fps)	mIoU
ASPP [14]	2048K	35.0	75.1
PPM [15]	261K	36.7	75.6
SPP (ours)	113K	41.0	75.4

5.11 Validating the upsampling capacity

This subsection presents validation and ablation experiments which provide further insight into the design of the upsampling path. For simplicity, we perform all experiments on the single-frame model.

Decoder width The number of feature maps along the upsampling path is the most important design choice of the decoder. We validate this hyper-parameter and report the results in Table 5.8. The results show that the model accuracy saturates at 128 dimensions. Consequently, we pick this value as a sensible speed-accuracy trade-off in all other experiments.

Table 5.8: Validation of the number of feature maps in the upsampling path of a single-frame architecture. The models were trained on Cityscapes train subset at 512×1024 while the evaluation is performed on Cityscapes val. All models use ImageNet initialization.

model					
ResNet-18 SPP	upsampling features	64	128	192	256
	mIoU	69.50	70.35	70.26	70.63

Lateral connections To demonstrate the importance of lateral connections between the encoder and the decoder, we train a single scale model without lateral connections. For this experiment, we discard the skip connections and elementwise summation in the upsampling modules. Such design decreases the validation accuracy on full-resolution Cityscapes from 75.4% to 72.5%.

Lighter upsampling We replace the 3×3 convolution of the upsampling module with a variety of lighter alternatives, while keeping the same training specification. Table 5.9 reports validation results for our single-scale model with ResNet-18 encoder on full-resolution Cityscapes images.

Table 5.10 validates the same lightweight upsampling alternatives for the MobileNet V2 encoder on CamVid resolution. The models were trained on CamVid train and the reported results are on CamVid val.

Tables 5.9 and 5.10 show that most lightweight upsampling techniques cause a drop in validation accuracy. Interestingly, the grouped convolution outperforms the standard convolution both in terms of accuracy and inference speed. Note that we nevertheless use standard convolution in all previous results since the advantage is insufficient to warrant repeating all experiments.

Table 5.9: Validation of convolution alternatives in the upsampling path of a single-scale model with a ResNet-18 encoder. The GFLOP column shows the number of floating point operations for the full Cityscapes resolution, i.e. 1024×2048. The *params* column displays the number of parameters in the decoder path (without ImageNet pre-trained parameters).

Upsampling alternative	mIoU	GFLOP	speed (fps)	params
grouped conv (8 groups)	**75.7**	84.1	**42.9**	252K
depthwise separable	73.8	84.0	43.9	250K
inverted residual	74.9	92.7	38.7	405K
1×1 conv	74.2	83.8	44.5	247K
3×3 conv	75.4	104	41.0	636K

Table 5.10: Evaluation of convolution alternatives in the upsampling path of a model with MobileNet V2 encoder. The *GFLOP* column shows the number of floating point operations when the input image is full CamVid resolution, i.e. 960×720. The *params* column displays the number of parameters in the decoder path without ImageNet pre-trained parameters.

Upsampling alternative	mIoU	GFLOP	speed (fps)	params
grouped conv(8 groups)	**72.8**	5.9	122.0	180K
depthwise separable	71.1	5.8	122.6	178K
inverted residual	72.0	8.6	108.1	332K
1×1 conv	71.7	5.7	127.9	174K
3×3 conv	72.1	13.1	116.0	567K

5.12 Validating the pyramid encoder

Table 5.11 investigates the impact of the number of pyramid levels to the Cityscapes validation accuracy. The table shows that an image pyramid with 3 levels yields the best validation performance (76.4% mIoU). Introducing the fourth level decreases validation accuracy to 76.0% mIoU. We believe that this performance decline indicates overfitting caused by a greater number of random parameters. The table also suggests that pyramids with one or two levels can not correctly recognize large objects due to insufficient effective receptive field as revealed in subsection 5.14.

A pyramidal fusion model without parameter sharing in the recognition encoder achieves around 70% mIoU on Cityscapes val in our standard training setup. This is almost 6pp less than the correponding model with shared parameters across the resolution pyramid. This suggests that parameter sharing is a very important ingredient of our pyramidal fusion architecture.

5.13 Improving the pyramidal fusion with boundary-aware loss

Our preliminary experiments with more than two levels of the pyramid and standard cross-entropy loss achieved no improvements. A closer inspection diagnosed inaccurate segmentation on distant objects [40], which suggested poor performance at semantic boundaries. When we introduced the boundary-aware loss (4.4), Cityscapes val mIoU rose from 75.5% to 76.4% for the 3-level ResNet-18 pyramidal fusion model. Interestingly, vanilla focal loss brings no improvement to any of our models, boundary-aware loss brings no improvement to the single-scale model, while boundary-aware loss without the focal loss component brings no improvement over the standard cross-entropy to our pyramid models. Figure 5.9 demonstrates typical contribution of the boundary-aware loss to the three-level pyramidal fusion model.

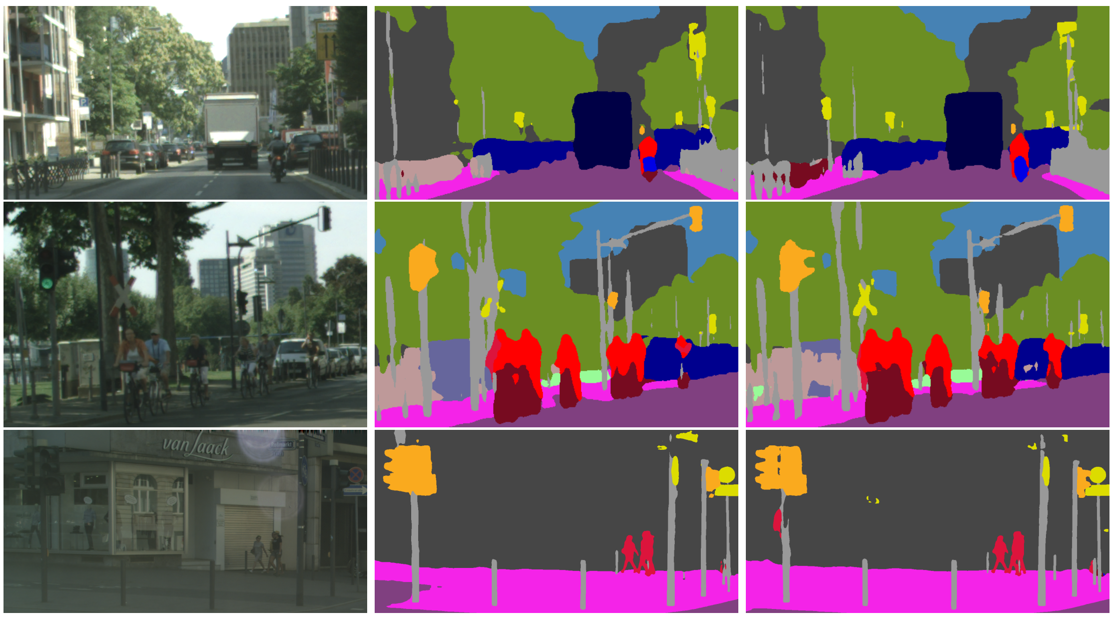

Figure 5.9: Situations where boundary-aware loss outperforms cross-entropy. Column 1 displays images from Cityscapes val, whereas columns 2 and 3 show segmentation outputs for pyramidal fusion models trained with cross-entropy and boundary-aware loss, respectively. Thin, distant or small objects are consistently better segmented when training with boundary-aware loss. Examples include the motorcycle rider and traffic poles (first row), cyclists and traffic signs (middle row), as well as sidewalk and traffic lights (bottom row). Credit: Oršić and Šegvić [41].

5.14 Interpreting the operation of the presented models

This subsection provides additional insight by interpreting and explaining decisions of our models [113]. In particular, we illustrate effects of pyramidal fusion and ImageNet initialization by leveraging attribution and ablation techniques. These experiments allow a human observer to explain outcomes of the inference in particular pixels, and to understand emergence of model decisions and learning dynamics.

Effective receptive field Effective receptive field (ERF) reveals the breadth of the context used to bring semantic predictions. We evaluate ERF as follows [114]. Firstly, we determine the partial derivative $\frac{\partial y_i}{\partial \mathbf{X}}$ where $\mathbf{y}$ are the logits of a given pixel, and $i = \arg\max(\mathbf{y})$. Secondly, we find top 100 thousand coordinates with the largest magnitude of the gradient $\frac{\partial y_i}{\partial \mathbf{X}}$. We express pixel-level ERF as standard deviations of these coordinates (in pixels) along the two image axes. Finally, we approximate the model-wide ERF as a mean pixel-level ERF in central pixels of all images from Cityscapes val. Table 5.11 presents the resulting ERF values for five of our models. We notice from these experiments that a larger ERF always leads to higher accuracy. This is intuitively clear, since context is the most valuable cue for classifying pixels in regions without discriminative local features. We note that the 4-level pyramid learns a smaller ERF and incurs a mIoU decline of 0.4pp with respect to the 3-level pyramid. This suggests that the model was unable to exploit features from the level-4 input.

Table 5.11: Model-wide effective receptive field (ERF) is strongly correlated with mIoU accuracy across five of our models. The models were trained on Cityscapes train, while mIoU and ERF were evaluated on Cityscapes val. We have used full resolution images.

model	ERF horizontal	ERF vertical	mIoU
RN-18 SPP	127.5	114.9	75.4
RN-18 1lvl	91.9	92.9	72.6
RN-18 2lvl	107.3	100.6	75.0
RN-18 3lvl	**133.1**	**115.1**	**76.4**
RN-18 4lvl	127.0	112.0	76.0

Removing residual units at test time This subsection evaluates resilience of our models to deleting k out of n residual units [63]. We proceed by randomly sampling $\min(40, \binom{n}{k})$ unique configurations with $n-k$ residual units. For the single-scale ResNet-18 model, $n = 8$. For the 3-level pyramidal fusion model, $n = 3 \times 8 = 24$. This means that we delete only one residual unit in both cases. Figure 5.10 shows that the pyramidal fusion model exhibits better deletion tolerance even when we remove three times as much residual units than in the single-scale case.

This suggests that the decision process in a pyramidal fusion model is much more distributed than than in the single-frame model. We believe that such ensemble-like behaviour positively affects the generalization ability of the pyramidal fusion architecture.

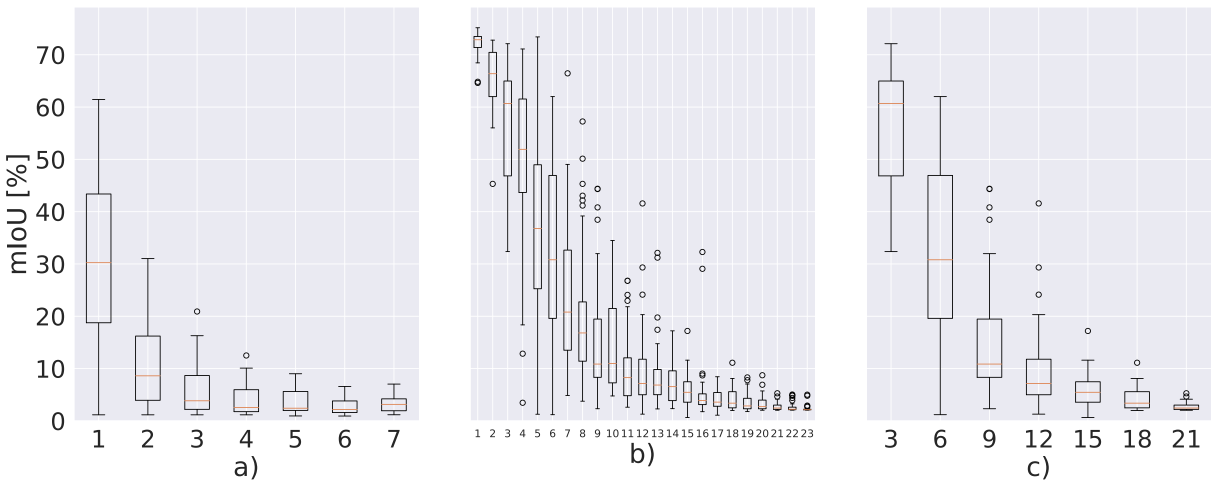

Figure 5.10: Effects of deleting residual units from the encoder at test time. The two box plots demonstrate how Cityscapes validation mIoU drops when residual units are omitted. The x-axis displays the number of deleted residual units. We show results for the single-scale model (a), and the pyramidal fusion model (b and c). The final plot (c) displays every third entry from the middle plot (b). Credit: Oršić and Šegvić [41].

Which skip connections contribute most gradient during training We study the relative importance of connections between the downsampling encoder and the upsampling decoder by performing the following experiments. First, we perform a forward pass on the input image. Then we modify the backward pass in a way to propagate gradients only through a chosen connection between the encoder and the decoder. We reference these connections by the subsampling factor from `4x` to `32x` (single-scale) or `128x` (pyramidal fusion). We propagate the gradients all the way to the input image ($\frac{\partial L}{\partial I}$) and to the parameters of the first convolutional layer ($\frac{\partial L}{\partial W^{conv1}}$). Figure 10 shows the results for the single-scale model (left) and the 3-level pyramidal fusion model (right). Unlike the single scale model, the pyramidal fusion favours the gradient flow through all connections. We notice that connections with most residual paths (`16x` and `32x`) contribute most gradient, which suggests that pyramidal fusion leads to more intensive learning.

Impact of ImageNet initialization Table 5.12 presents ablation experiments which explore benefits of ImageNet pre-training. In each experiment, we use ImageNet initialization in one additional residual block to find out where this regularization helps the most. In each case we train SwiftNet-RN18 on full resolution Cityscapes train and report mIoU on Cityscapes val . Interestingly, the obtained results show that ImageNet pretraining benefits all processing blocks

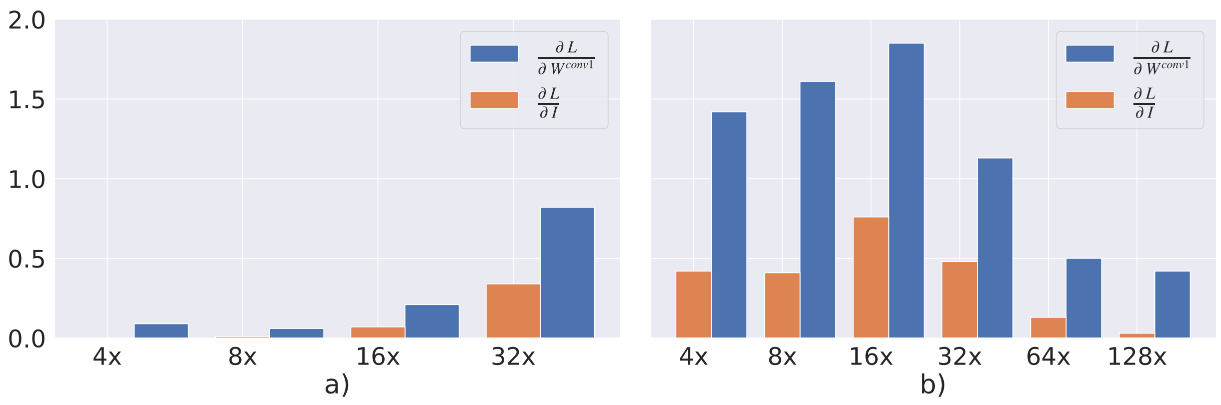

Figure 5.11: Magnitude of gradients incurred by each skip connection for the single scale (a) and the pyramidal fusion model (b). We calculate gradients w.r.t. the input image ($\frac{\partial L}{\partial I}$) as well as gradients w.r.t. the convolutional kernels in the first layer ($\frac{\partial L}{\partial W^{conv1}}$). The horizontal axis represents the stride of the skip connection w.r.t. the input image. Maximum stride in both graphs shows the decoder input and not the actual skip connection. Credit: Oršić and Šegvić [41].

of the backbone. We believe that early layers are positively affected due to clear utility of early features, while the later layers likely profit due to decreased opportunity to overfit.

Table 5.12: Impact of partial ImageNet pre-training of the single-scale model to the accuracy on Cityscapes val. Row *modules* denotes which residual blocks were initialized on ImageNet (cumulative from left to right). Row *params* displays the total number of parameters which were initialized on ImageNet. All models were trained at full resolution.

modules	none	+stem	+RB 1	+RB 2	+RB 3	+RB 4
params	—	9K	157K	683K	2.8M	11.2M
mIoU	68.50	69.21	69.98	71.74	73.08	**75.35**

Ensembling pyramidal fusion and single scale models We compare the single scale SPP model with the pyramid model by evaluating the following two ensembles on Cityscapes val: i) one SPP model and one pyramid model, and ii) two SPP models. Table 5.13 shows that most improvement over the single SPP model is achieved with a heterogeneous ensemble. This indicates that the two approaches learn different representations. This also supports results from Table 5.11 which indicate that the two models have very different effective receptive fields.

Table 5.13: Experiments of segmentation mIoU of ensembled models on Cityscapes val. The combination of one single scale(SPP) and one pyramidal fusion model surpasses the ensemble of two single scale models models.

models	first	second	ensemble
SPP1 + SPP2	75.4	75.4	76.5
pyramid + SPP1	76.5	75.4	78.0
pyramid + SPP2	76.5	75.4	**78.1**

Chapter 6

Participation in Robust Vision Challenge 2020

Deep models achieve outstanding recognition quality when applied to a single domain. However, when applied across multiple domains, they often fail to conceive whether unknown concepts are observed. Instead, single-domain models usually produce random or overconfident erroneous outputs as response to input data from unseen domains. The problem is two fold. Firstly, models trained on a single domain output a probability across domain classes only. Secondly, single-domain models are not trained to handle cases in which novel concepts are presented.

Estimating prediction uncertainty is important in real-world applications. This is especially beneficial in dense prediction tasks such as semantic segmentation. Using information about which pixels have confident predictions improves method robustness. Estimating model uncertainty is related to outlier detection which is an important research area [115, 116].

6.1 Multi domain semantic segmentation benchmark

Model robustness can be assessed by applying a single model to multiple domains. This is the main proposition of Robust Vision Challenge (RVC), a biannual computer vision competition. RVC evaluates model quality by evaluating a single model on a collection of public datasets with vastly different characteristics. Semantic segmentation datasets are described in Table 6.1. In total, there are seven datasets with 93,369 labeled training images. The test set consists of 15,524 unlabeled images which were not observed during model training. ADE20k contains rather small indoor and outdoor images, and has the most classes. ScanNet contains interior images with very noisy labels. Cityscapes, KITTI, Vistas, VIPER, and WildDash 2 contain road-driving images. Cityscapes contains images from western Europe taken with the same camera in fine weather. Vistas contains crowd-sourced images across the globe in all kinds

Dataset	content	# images	# classes	resolution
ADE20K	photos	22210	150 / 150	460±154
Cityscapes	driving	3475	28 / 19	1448±0
KITTI	driving	200	28 / 19	682±1
VIPER	artificial	18326	32 / 19	1440±0
ScanNet	interior	24902	40 / 20	1109±78
Vistas	driving	20000	65 / 65	2908±608
WildDash 2	driving	4256	26 / 20	1440±0

Table 6.1: Summary of the seven datasets from the RVC 2020 collection for semantic segmentation. The columns correspond to the total number of annotated non-test images (# images), the total number of training and test classes (# classes), as well as the mean and standard deviation of the square root of the number of pixels ($\sqrt{HW}$) across the training split (resolution).

of weather. WildDash 2 collects hand-picked images according to a system of hazards [117]. VIPER contains images generated by playing a computer game.

Each of the seven RVC2020 datasets defines class taxonomies which were independently created. This causes situations where a naive concatenation would introduce contradictory labels. Let us demonstrate this on the concept of a pickup truck (a small closed cabin truck with open cargo area) visualized in 6.1. Pickups are labeled as `truck` in VIPER, `van` in ADE20k and `car` in Vistas. Each dataset specific class also includes concepts of other driving vehicles. For example, Vistas `car` includes ordinary cars but does not include large trucks which are present in VIPER `truck`. This example demonstrates how discriminative learning on such overlapping classes causes label noise.

We addressed the issue of overlapping taxonomies by introducing a loss function which supports training over such taxonomies. The loss is expressed as negative log-likelihood of summed probabilities. Conveniently, we named the loss function NLL+. Furthermore, we created a discrete mapping from dataset-specific taxonomies into a set of universal classes. This requires understanding of semantic concepts covered by each dataset-specific class. In the case of seven RVC2020 taxonomies, mapping was created within one working day. The procedure results in one-to-many mappings. This is aligned with NLL+ which supports training with multiple ground-truth classes. Interestingly, NLL+ was previously used to enable training on partial labels [118].

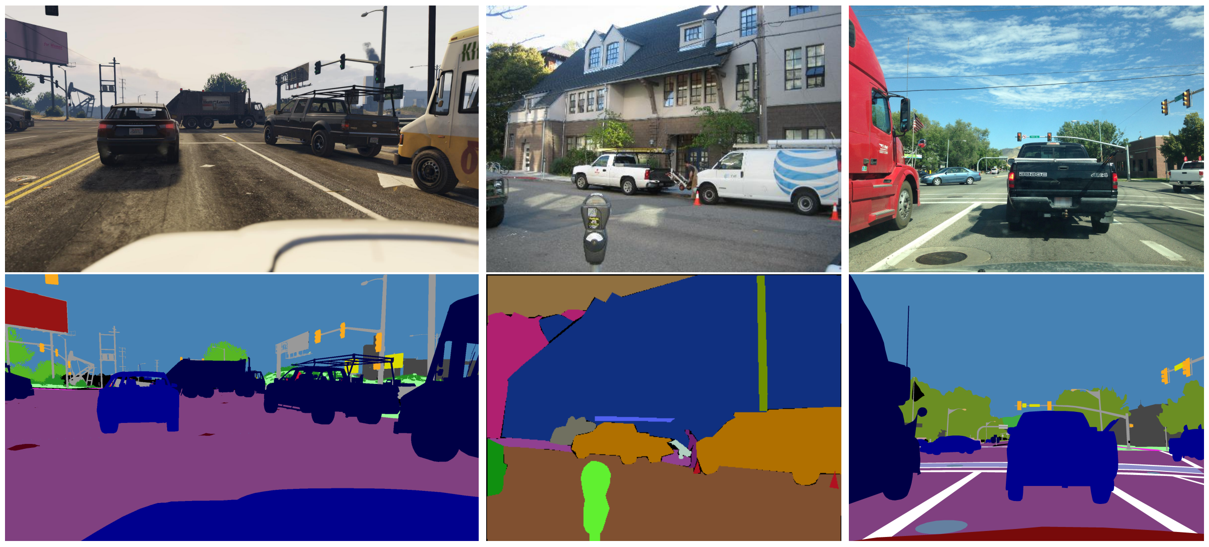

Figure 6.1: Example of inharmonious taxonomies for multi-domain semantic segmentation. Pickups are labeled as class `truck` in VIPER [119] (left), class `van` in Ade20k [100] (middle) and class `car` in Vistas [24] (right). We resolve the class overlap by learning on partial labels [118].

6.2 Pyramidal fusion for multi domain semantic segmentation

Our submission used a modified version of pyramidal fusion. A real-time capable model would not be able to fit all training data due to small capacity. Therefore, we increased the model capacity in two ways. First, we introduced a bigger encoder, namely ResNet-152 [96]. Second, we increased the number of feature maps in the decoder threefold. The resulting model had sufficient capacity enabled by pyramidal fusion and enhanced encoder-decoder configuration. We compare computational characteristics of this model to other widely used architectures in Table 6.2. Note how the large number of logits (192) introduces great memory requirements during model training. Logits are upsampled to label resolution when applying the loss. For example, calculating the loss on logits with 192 classes and 1MPx resolution allocates 1.5G of memory (this calculation disregards memory required by the segmentation model). We identify this as the main bottleneck for training segmentation models in datasets with a large number of classes.

6.3 Competition results

We present our submission to the RVC 2020 semantic segmentation benchmark. The submission is trained on 6 Tesla V100 GPUs with 32GB RAM. We train exclusively on the seven RVC 2020 datasets. We compose mini-batches with roulette wheel sampling [102]. We favour fair representation of classes within datasets by encouraging sampling of images with multiple class

Model	M-Adds	Params	Memory
DLv3-RN101	967G	58.7M	2.4G / 10.6G
PSPNet-RN50	752G	46.8M	1.3G / 8.7G
HN-OCR-W48	649G	70.3M	1.5G / 7.5G
PDL-HN48+	359G	68.9M	1.2G / 8.2G
SNpyrx8-RN152	338G	60.3M	1.5G / 8.2G
SNpyrx8-RN18	68G	12.3M	1.1G / 3.5G

Table 6.2: Computational complexity, parameter count and memory footprint (eval/train) for prominent semantic segmentation models. We assume 1MPx input, 192 logits, and batch size 1. SNpyr stands for pyramidal SwiftNet [41]. PDL denotes Panoptic DeepLab [120] without an instance decoder. RN, X and HN stand for ResNet [42], Xception [121] and HRNet [122], respectively.

Model	ADE20k	Cityscapes	KITTI	Vistas	ScanNet	VIPER	WildDash 2
MSeg1080	**33.2**	**80.7**	62.6	34.2	48.5	40.7	35.2
seamseg bl	-	-	-	-	-	36.1	37.9
EffPS_b1bs4	-	-	-	-	-	52.0	32.2
SNp_RN152	31.1	74.7	**63.9**	**40.4**	**54.6**	**62.5**	**45.4**

Table 6.3: Performance evaluation on the RVC 2020 benchmark collection. We compare mIoU accuracy of the concurrent work (top) with our model (bottom).

instances and images with rare classes. We attempt to alleviate noisy ScanNet labels by setting the boundary modulation to 1 (minimum) for all ScanNet crops. Evaluation on RVC 2020 requires performing inference on 15,524 images with varying resolution, which requires around 50 GPU hours. Competition results are presented in Table 6.3.

Our fully convolutional model receives a colour image on input and produces dense predictions into 192 universal classes. We produce the logits at full resolution by 8$\times$ bilinear upsampling. We recover dataset-specific predictions (these are needed for training and benchmarking) by summing softmax probabilities of all universal classes which map to the particular dataset class. We recover the probability of the void class by summing probabilities of all universal classes which are not covered by the dataset-specific taxonomy. This particular choice enables the detection of out-of-distribution pixels on WildDash 2 test.

Our submission trains a SNpyr-RN152 model without batchnorm syncronization. We update population statistics on only one GPU and perform model updates by accumulating gradients from all GPUs. We favour smooth evolution of batchnorm statistics and speed-up the training by gradually increasing the crop size according to the schedule from Table 6.4. At epoch 50, we freeze all batch normalization layers, reset the gradient moments used by Adam

Epochs	crop size	batch size	jitter range	speed
0 – 15	384	6×16	0.75 – 1.33	45 fps
16 – 31	512	6×8	0.60 – 1.67	27 fps
32 – 49	768	6×4	0.50 – 2.00	14 fps
50 – 52	1024	6×2	0.40 – 2.50	9 fps

Table 6.4: Mini-batch configuration schedule across the training epochs for the SNpyr-RN152 submission. The columns show the square crop size, the batch size, the range of uniform scale jittering and the training speed in crops per second.

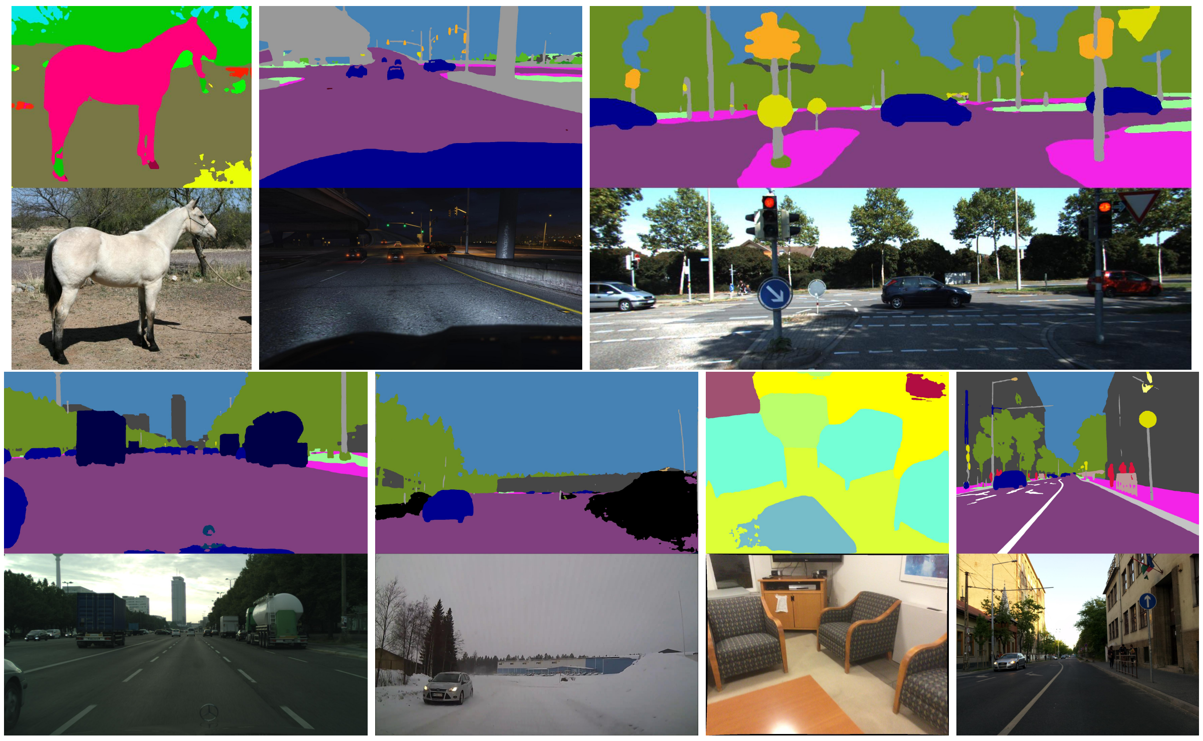

Figure 6.2: Qualitative performance of SNPyr_RN152 on RVC 2020 test. Rows 1 and 3 show input images, while rows 2 and 4 show the model predictions. Images belong to (top to bottom, left to right): ADE, Viper, Kitti, Cityscapes, WildDash, ScanNet and Vistas.

and train for 3 more epochs. The training involves 140k iterations, which took around 4 days on our hardware. We evaluate on the original resolution and two additional scales.

Our submission won the 2020 edition of Robust Vision Challenge. It achieved the best overall score in four out of seven datasets. On WildDash 2, we achieved best overall mIoU as well as best score in detecting negative (outlier) pixels. This result highlights the benefits of training over a universal taxonomy using NLL+. Input images and predicted segmentation maps are visualized in Figure 6.2 (one sample for each dataset). Note that WildDash 2 example contains predictions colored black. This corresponds to predicted outlier regions.

Robust Vision Challenge is an important benchmark for most important computer vision tasks. Magnitude of datasets in the multi domain setting test the scalability of current approaches. The results suggest a gap between single domain public benchmarks and real-world applications as RVC2020 winners rarely deliver SotA single dataset results. Effectiveness of pyramidal fusion was demonstrated on RVC2020 semantic segmentation benchmarks. Participating in the challenge highlighted how this efficient approach works well when applied in a setting requiring large model capacity.

Chapter 7

Conclusion

Real-time performance is a very important trait of semantic segmentation models aiming at applications which require low latency and immediate decisioning. Most previous work in the field involves custom convolutional encoders trained from scratch, and decoders without lateral skip-connections. However, these approaches are unable to match accuracy provided by backbones with competitive ImageNet performance capacity. Our single-scale model shows that some of the accuracy lost due to low backbone capacity can be reclaimed by extending the receptive range. This is especially effective in the case of ImageNet-pre-trained models which do not pay attention to the wider context, since typical ImageNet training setups involve a very small resolution.

We have revisited a principled recognition approach based on shared encoders across a resolution pyramid due to clear potential for scale covariance, large receptive range, and knowledge transfer from ImageNet. We have shown that this potential materializes under the following two requirements. First, representations from *all* encoder blocks and *all* pyramid levels have to be fused within a ladder-style decoder. We term such wiring as pyramidal fusion. Second, the compound model has to be trained with a boundary-aware loss. Interpretation of the resulting models shows that the learned receptive ranges are indeed large, and suggests that pyramidal fusion succeeds due to efficient learning and ensemble-like behaviour.

We have provided a detailed analysis of prediction accuracy and processing time for models based on ResNet-18 and MobileNet V2 backbones. Our models display competitive performance on CamVid, Vistas and ADE20k. Our best Cityscapes test submission achieves 76.4% mIoU at 34 Hz on a GTX1080Ti when processing images of 1024$\times$2048 pixels. To the best of our knowledge, this result outperforms all previous semantic segmentation approaches aiming at real-time application. The source code is available at `https://github.com/orsic/swiftnet`.

Bibliography

[1] Girshick, R., Donahue, J., Darrell, T., Malik, J., “Rich feature hierarchies for accurate object detection and semantic segmentation”, in CVPR, 2014.

[2] He, K., Gkioxari, G., Doll á r, P., Girshick, R., “Mask R-CNN”, in ICCV, 2017.

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

[4] Zbontar, J., LeCun, Y. *et al.*, “Stereo matching by training a convolutional neural network to compare image patches.”, JMLR, 2016.

[5] Kendall, A., Martirosyan, H., Dasgupta, S., Henry, P., Kennedy, R., Bachrach, A., Bry, A., “End-to-end learning of geometry and context for deep stereo regression”, in CVPR, 2017.

[6] Dosovitskiy, A., Fischer, P., Ilg, E., Hausser, P., Hazirbas, C., Golkov, V., Van Der Smagt, P., Cremers, D., Brox, T., “Flownet: Learning optical flow with convolutional networks”, in ICCV, 2015.

[7] Sun, D., Yang, X., Liu, M.-Y., Kautz, J., “PWC - Net : CNNs for optical flow using pyramid, warping, and cost volume”, in CVPR, 2018.

[8] Geiger, A., Lenz, P., Stiller, C., Urtasun, R., “Vision meets robotics: The kitti dataset”, The International Journal of Robotics Research, 2013.

[9] Mayer, N., Ilg, E., Hausser, P., Fischer, P., Cremers, D., Dosovitskiy, A., Brox, T., “A large dataset to train convolutional networks for disparity, optical flow, and scene flow estimation”, in CVPR, 2016.

[10] Yang, G., Ramanan, D., “Volumetric correspondence networks for optical flow”, in Advances in neural information processing systems, 2019, pp. 794–805.

[11] Teed, Z., Deng, J., “Raft: Recurrent all-pairs field transforms for optical flow”, in ECCV, 2020.

[12] Doersch, C., Gupta, A., Efros, A. A., “Context as supervisory signal: Discovering objects with predictable context”, in ECCV, 2014, pp. 362–377.

[13] Kirillov, A., He, K., Girshick, R., Rother, C., Dollar, P., “Panoptic segmentation”, in CVPR, June 2019.

[14] Chen, L.-C., Zhu, Y., Papandreou, G., Schroff, F., Adam, H., “Encoder-decoder with atrous separable convolution for semantic image segmentation”, in ECCV, 2018, pp. 801–818.

[15] Zhao, H., Shi, J., Qi, X., Wang, X., Jia, J., “Pyramid scene parsing network”, in CVPR, 2017.

[16] Yang, M., Yu, K., Zhang, C., Li, Z., Yang, K., “DenseASPP for semantic segmentation in street scenes”, in CVPR, 2018, pp. 3684–3692.

[17] Romera, E., Alvarez, J. é. M., Bergasa, L. M., Arroyo, R., “Erfnet: Efficient residual factorized convnet for real-time semantic segmentation”, T - ITS, Vol. 19, No. 1, 2017, pp. 263–272.

[18] Zhao, H., Qi, X., Shen, X., Shi, J., Jia, J., “ICNet for real-time semantic segmentation on high-resolution images”, in ECCV, 2018, pp. 405–420.

[19] Mehta, S., Rastegari, M., Caspi, A., Shapiro, L. G., Hajishirzi, H., “ESPNet : Efficient spatial pyramid of dilated convolutions for semantic segmentation”, in ECCV, 2018, pp. 561–580.

[20] Siam, M., Gamal, M., Abdel-Razek, M., Yogamani, S., Jagersand, M., Zhang, H., Vallurupalli, N., Annamaneni, S., Varma, G., Jawahar, C. *et al.*, “A comparative study of real-time semantic segmentation for autonomous driving”, in CVPR Workshops, 2018, pp. 587–597.

[21] Oquab, M., é on Bottou, L., Laptev, I., Sivic, J., “Learning and transferring mid-level image representations using convolutional neural networks”, in CVPR, 2014, pp. 1717–1724.

[22] 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”, IJCV, Vol. 115, No. 3, 2015, pp. 211-252.

[23] Cordts, M., Omran, M., Ramos, S., Rehfeld, T., Enzweiler, M., Benenson, R., Franke, U., Roth, S., Schiele, B., “The cityscapes dataset for semantic urban scene understanding”, in CVPR, 2016.

[24] Neuhold, G., Ollmann, T., Rota Bulo, S., Kontschieder, P., “The mapillary vistas dataset for semantic understanding of street scenes”, in ICCV, 2017.

[25] Maggiori, E., Tarabalka, Y., Charpiat, G., Alliez, P., “Can semantic labeling methods generalize to any city? the inria aerial image labeling benchmark”, in 2017 IEEE International Geoscience and Remote Sensing Symposium (IGARSS). IEEE, 2017, pp. 3226–3229.

[26] Chebrolu, N., Lottes, P., Schaefer, A., Winterhalter, W., Burgard, W., Stachniss, C., “Agricultural robot dataset for plant classification, localization and mapping on sugar beet fields”, The International Journal of Robotics Research, Vol. 36, No. 10, 2017, pp. 1045–1052.

[27] Dai, A., Chang, A. X., Savva, M., Halber, M., Funkhouser, T., Nießner, M., “Scannet: Richly-annotated 3d reconstructions of indoor scenes”, in Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2017, pp. 5828–5839.

[28] Farabet, C., Couprie, C., Najman, L., LeCun, Y., “Learning hierarchical features for scene labeling”, PAMI, Vol. 35, No. 8, 2012, pp. 1915–1929.

[29] Yu, F., Koltun, V., “Multi-scale context aggregation by dilated convolutions”, in ICLR, 2016.

[30] Rasmus, A., Berglund, M., Honkala, M., Valpola, H., Raiko, T., “Semi-supervised learning with ladder networks”, in NeurIPS, 2015, pp. 3546–3554.

[31] Ronneberger, O., Fischer, P., Brox, T., “U-net: Convolutional networks for biomedical image segmentation”, in MICCAI. Springer, 2015, pp. 234–241.

[32] Lin, T. . Y., á r, P. D., Girshick, R. B., He, K., Hariharan, B., Belongie, S. J., “Feature pyramid networks for object detection”, in CVPR, 2017, pp. 936–944.

[33] Krešo, I., Šegvić, S., Krapac, J., “Ladder-style DenseNets for semantic segmentation of large natural images”, in ICCVW, 2017, pp. 238–245.

[34] Gao, L., Zhou, Z., Shen, H. T., Song, J., “Bottom-up and top-down: Bidirectional additive net for edge detection”, in Proceedings of the Twenty-Ninth International Joint Conference on Artificial Intelligence, IJCAI 2020 [scheduled for July 2020, Yokohama, Japan, postponed due to the Corona pandemic], 2020, pp. 594–600.

[35] Zhen, M., Wang, J., Zhou, L., Fang, T., Quan, L., “Learning fully dense neural networks for image semantic segmentation”, in AAAI, 2019.

[36] Lowe, D. G., “Distinctive image features from scale-invariant keypoints”, International journal of computer vision, Vol. 60, No. 2, 2004, pp. 91–110.

[37] Lindeberg, T., Scale-space theory in computer vision. Springer Science & Business Media, 2013, Vol. 256.

[38] Krešo, I., Čaušević, D., Krapac, J., Šegvić, S., “Convolutional scale invariance for semantic segmentation”, in GCPR, 2016, pp. 64–75.

[39] Singh, B., Davis, L. S., “An analysis of scale invariance in object detection - SNIP”, in CVPR, 2018, pp. 3578–3587.

[40] Oršić, M., Krešo, I., Bevandić, P., Šegvić, S., “In defense of pre-trained ImageNet architectures for real-time semantic segmentation of road-driving images”, in Proceedings of the IEEE conference on computer vision and pattern recognition, 2019, pp. 12 607–12 616.

[41] Oršić, M., Šegvić, S., “Efficient semantic segmentation with pyramidal fusion”, Pattern Recognition, 2020, p. 107611.

[42] He, K., Zhang, X., Ren, S., Sun, J., “Deep residual learning for image recognition”, in CVPR, 2016, pp. 770–778.

[43] Sandler, M., Howard, A., Zhu, M., Zhmoginov, A., Chen, L.-C., “MobileNetV2 : Inverted residuals and linear bottlenecks”, in CVPR, 2018.

[44] Krizhevsky, A., Sutskever, I., Hinton, G. E., “Imagenet classification with deep convolutional neural networks”, in NeurIPS, 2012.

[45] Deng, J., Dong, W., Socher, R., Li, L.-J., Li, K., Fei-Fei, L., “Imagenet: A large-scale hierarchical image database”, in 2009 IEEE conference on computer vision and pattern recognition. Ieee, 2009, pp. 248–255.

[46] Sánchez, J., Perronnin, F., “High-dimensional signature compression for large-scale image classification”, in CVPR 2011. IEEE, 2011, pp. 1665–1672.

[47] Simonyan, K., Zisserman, A., “Very deep convolutional networks for large-scale image recognition”, in 3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings, 2015.

[48] LeCun, Y., Bottou, L. é. o., Bengio, Y., Haffner, P., “Gradient-based learning applied to document recognition”, Proceedings of the IEEE, Vol. 86, No. 11, 1998, pp. 2278–2324.

[49] Netzer, Y., Wang, T., Coates, A., Bissacco, A., Wu, B., Ng, A. Y., “Reading digits in natural images with unsupervised feature learning”, 2011.

[50] Krizhevsky, A., Hinton, G. *et al.*, “Learning multiple layers of features from tiny images”, 2009.

[51] Kuznetsova, A., Rom, H., Alldrin, N., Uijlings, J., Krasin, I., Pont-Tuset, J., Kamali, S., Popov, S., Malloci, M., Kolesnikov, A., Duerig, T., Ferrari, V., “The open images dataset v4: Unified image classification, object detection, and visual relationship detection at scale”, IJCV, 2020.

[52] Ren, S., He, K., Girshick, R., Sun, J., “Faster r-cnn: Towards real-time object detection with region proposal networks”, in NeurIPS, 2015, pp. 91–99.

[53] Fukushima, K., “A self-organizing neural network model for a mechanism of pattern recognition unaffected by shift in position”, Biol. Cybern., Vol. 36, 1980, pp. 193–202.

[54] Hirose, Y., Yamashita, K., Hijiya, S., “Back-propagation algorithm which varies the number of hidden units”, Neural networks, Vol. 4, No. 1, 1991, pp. 61–66.

[55] Nair, V., Hinton, G. E., “Rectified linear units improve restricted boltzmann machines”, in Proceedings of the 27th international conference on machine learning (ICML-10), 2010, pp. 807–814.

[56] Ioffe, S., Szegedy, C., “Batch normalization: Accelerating deep network training by reducing internal covariate shift”, in ICML, 2015, pp. 448–456.

[57] Srivastava, N., Hinton, G., Krizhevsky, A., Sutskever, I., Salakhutdinov, R., “Dropout: a simple way to prevent neural networks from overfitting”, The journal of machine learning research, Vol. 15, No. 1, 2014, pp. 1929–1958.

[58] Lavin, A., Gray, S., “Fast algorithms for convolutional neural networks”, in Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2016, pp. 4013–4021.

[59] Behrmann, J., Grathwohl, W., Chen, R. T., Duvenaud, D., Jacobsen, J.-H., “Invertible residual networks”, in International Conference on Machine Learning. PMLR, 2019, pp. 573–582.

[60] Srivastava, R. K., Greff, K., Schmidhuber, J., “Highway networks”, CoRR, Vol. abs/1505.00387, 2015.

[61] Zilly, J. G., Srivastava, R. K., Koutnık, J., Schmidhuber, J., “Recurrent highway networks”, in International Conference on Machine Learning. PMLR, 2017, pp. 4189–4198.

[62] Oršić, M., "Učenje korespondencijske metrike za gustu stereoskopsku rekonstrukciju", 2017.

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

[64] Goyal, P., Dollár, P., Girshick, R., Noordhuis, P., Wesolowski, L., Kyrola, A., Tulloch, A., Jia, Y., He, K., "Accurate, large minibatch sgd: Training imagenet in 1 hour", arXiv preprint arXiv:1706.02677, 2017.

[65] Huang, G., Liu, Z., Pleiss, G., Van Der Maaten, L., Weinberger, K., "Convolutional networks with dense connectivity", IEEE transactions on pattern analysis and machine intelligence, 2019.

[66] Krešo, I., Krapac, J., Šegvić, S., "Efficient ladder-style DenseNets for semantic segmentation of large images", IEEE Transactions on Intelligent Transportation Systems, 2020.

[67] Kolesnikov, A., Zhai, X., Beyer, L., "Revisiting self-supervised visual representation learning", in Proceedings of the IEEE conference on Computer Vision and Pattern Recognition, 2019, pp. 1920–1929.

[68] Zhai, X., Oliver, A., Kolesnikov, A., Beyer, L., "S4l: Self-supervised semi-supervised learning", in Proceedings of the IEEE international conference on computer vision, 2019, pp. 1476–1485.

[69] Oršić, M., Bevandić, P., Grubišić, I., Šarić, J., Šegvić, S., "Multi-domain semantic segmentation with pyramidal fusion", arXiv preprint arXiv:2009.01636, 2020.

[70] Šegvić, S., "Deep learning, lectures", 2018.

[71] Zhang, X., Zhou, X., Lin, M., Sun, J., "Shufflenet: An extremely efficient convolutional neural network for mobile devices", in CVPR, 2018, pp. 6848–6856.

[72] Huang, G., Liu, S., Van der Maaten, L., Weinberger, K. Q., "Condensenet: An efficient densenet using learned group convolutions", in CVPR, 2018, pp. 2752–2761.

[73] Zoph, B., Vasudevan, V., Shlens, J., Le, Q. V., "Learning transferable architectures for scalable image recognition", in CVPR, 2018, pp. 8697–8710.

[74] Tan, M., Le, Q. V., "Efficientnet: Rethinking model scaling for convolutional neural networks", in Proceedings of the 36th International Conference on Machine Learning, ICML 2019, 9-15 June 2019, Long Beach, California, USA, 2019, pp. 6105–6114.

[75] Chen, L.-C., Yang, Y., Wang, J., Xu, W., Yuille, A. L., “Attention to scale: Scale-aware semantic image segmentation”, in CVPR, 2016, pp. 3640–3649.

[76] 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, pp. 2481–2495.

[77] Yu, F., Koltun, V., Funkhouser, T., “Dilated residual networks”, in CVPR, 2017, pp. 472–480.

[78] He, K., Zhang, X., Ren, S., Sun, J., “Spatial pyramid pooling in deep convolutional networks for visual recognition”, PAMI, Vol. 37, No. 9, 2015.

[79] Lazebnik, S., Schmid, C., Ponce, J., “Beyond bags of features: Spatial pyramid matching for recognizing natural scene categories”, in CVPR, Vol. 2. IEEE, 2006.

[80] Hu, J., Shen, L., Sun, G., “Squeeze-and-excitation networks”, in CVPR, June 2018.

[81] Mazzini, D., “Guided upsampling network for real-time semantic segmentation”, in BMVC, 2018, p. 117.

[82] Qin, H., Gong, R., Liu, X., Bai, X., Song, J., Sebe, N., “Binary neural networks: A survey”, Pattern Recognit., Vol. 105, 2020, p. 107281.

[83] Frankle, J., Carbin, M., “The lottery ticket hypothesis: Finding sparse, trainable neural networks”, in 7th International Conference on Learning Representations, ICLR 2019, New Orleans, LA, USA, May 6-9, 2019, 2019.

[84] Sifre, L., é phane Mallat, S., “Rigid-motion scattering for texture classification”, CoRR, 2014.

[85] Wang, M., Liu, B., Foroosh, H., “Factorized convolutional neural networks.”, in ICCV Workshops, 2017, pp. 545–553.

[86] Huang, G., Liu, Z., Van Der Maaten, L., Weinberger, K. Q., “Densely connected convolutional networks”, in CVPR, 2017.

[87] Paszke, A., Chaurasia, A., Kim, S., Culurciello, E., “Enet: A deep neural network architecture for real-time semantic segmentation”, CoRR, 2016.

[88] Vallurupalli, N., Annamaneni, S., Varma, G., Jawahar, C., Mathew, M., Nagori, S., “Efficient semantic segmentation using gradual grouping”, in CVPR Workshops, 2018, pp. 598–606.

[89] Yang, K., Hu, X., Bergasa, L. M., Romera, E., Wang, K., “Pass: Panoramic annular semantic segmentation”, IEEE Transactions on Intelligent Transportation Systems, 2019.

[90] Wu, J., Wen, Z., Zhao, S., Huang, K., “Video semantic segmentation via feature propagation with holistic attention”, Pattern Recognition, 2020.

[91] Chaurasia, A., Culurciello, E., “LinkNet : Exploiting encoder representations for efficient semantic segmentation”, in VCIP, 2017, pp. 1–4.

[92] Nekrasov, V., Shen, C., Reid, I. D., “Light-weight refinenet for real-time semantic segmentation”, in BMVC, 2018, p. 125.

[93] Yang, K., Hu, X., Chen, H., Xiang, K., Wang, K., Stiefelhagen, R., “Ds-pass: Detail-sensitive panoramic annular semantic segmentation through swaftnet for surrounding sensing”, CoRR, 2019.

[94] Li, H., Xiong, P., Fan, H., Sun, J., “Dfanet: Deep feature aggregation for real-time semantic segmentation”, in CVPR, June 2019.

[95] Li, X., Zhou, Y., Pan, Z., Feng, J., “Partial order pruning: For best speed/accuracy trade-off in neural architecture search”, in CVPR, 2019, pp. 9145–9153.

[96] He, K., Zhang, X., Ren, S., Sun, J., “Identity mappings in deep residual networks”, in ECCV. Springer, 2016.

[97] Zhang, P., Liu, W., Wang, H., Lei, Y., Lu, H., “Deep gated attention networks for large-scale street-level scene segmentation”, Pattern Recognition, Vol. 88, 2019, pp. 702–714.

[98] Lin, T.-Y., Goyal, P., Girshick, R., He, K., Doll á r, P., “Focal loss for dense object detection”, in ICCV, 2017.

[99] Brostow, G. J., Fauqueur, J., Cipolla, R., “Semantic object classes in video: A high-definition ground truth database”, Pattern Recognition Letters, Vol. 30, No. 2, 2009, pp. 88–97.

[100] Zhou, B., Zhao, H., Puig, X., Fidler, S., Barriuso, A., Torralba, A., “Scene parsing through ade20k dataset”, in CVPR, 2017, pp. 633–641.

[101] Kingma, D. P., Ba, J., “Adam: A method for stochastic optimization”, in ICLR, 2015.

[102] Rota Bulò, S., Porzi, L., Kontschieder, P., “In-place activated batchnorm for memory-optimized training of dnns”, in CVPR, 2018, pp. 5639–5647.

[103] Li, Y., Yuan, L., Vasconcelos, N., “Bidirectional learning for domain adaptation of semantic segmentation”, in CVPR, June 2019.

[104] Cheng, B., Collins, M. D., Zhu, Y., Liu, T., Huang, T. S., Adam, H., Chen, L.-C., “Panoptic- DeepLab : A simple, strong, and fast baseline for bottom-up panoptic segmentation”, CoRR, 2019.

[105] Lin, G., Milan, A., Shen, C., Reid, I., “Refinenet: Multi-path refinement networks for high-resolution semantic segmentation”, in CVPR, 2017, pp. 1925–1934.

[106] Zhang, R., Yang, W., Peng, Z., Wei, P., Wang, X., Lin, L., “Progressively diffused networks for semantic visual parsing”, Pattern Recognition, Vol. 90, 2019, pp. 78–86.

[107] Fu, J., Liu, J., Wang, Y., Li, Y., Bao, Y., Tang, J., Lu, H., “Adaptive context network for scene parsing”, in ICCV, 2019, pp. 6748–6757.

[108] Wu, Z., Shen, C., Van Den Hengel, A., “Wider or deeper: Revisiting the resnet model for visual recognition”, Pattern Recognition, Vol. 90, 2019, pp. 119–133.

[109] Fu, J., Liu, J., Li, Y., Bao, Y., Yan, W., Fang, Z., Lu, H., “Contextual deconvolution network for semantic segmentation”, Pattern Recognition, Vol. 101, 2020, p. 107152.

[110] Huang, Z., Wang, X., Huang, L., Huang, C., Wei, Y., Liu, W., “CCNet : Criss-cross attention for semantic segmentation”, in ICCV, 2019, pp. 603–612.

[111] Bayer, B. E., “Color imaging array”, uS Patent 3,971,065. 1976.

[112] Zendel, O., Murschitz, M., Zeilinger, M., Steininger, D., Abbasi, S., Beleznai, C., “Railsem19: A dataset for semantic rail scene understanding”, in Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops, 2019, pp. 0–0.

[113] Miller, T., “Explanation in artificial intelligence: Insights from the social sciences”, Artificial Intelligence, Vol. 267, 2019, pp. 1–38.

[114] Luo, W., Li, Y., Urtasun, R., Zemel, R., “Understanding the effective receptive field in deep convolutional neural networks”, in NeurIPS, 2016.

[115] Bevandić, P., Krešo, I., Oršić, M., Šegvić, S., “Discriminative out-of-distribution detection for semantic segmentation”, arXiv preprint arXiv:1808.07703, 2018.

[116] Bevandić, P., Krešo, I., Oršić, M., Šegvić, S., “Simultaneous semantic segmentation and outlier detection in presence of domain shift”, in German Conference on Pattern Recognition. Springer, 2019, pp. 33–47.

[117] Zendel, O., Honauer, K., Murschitz, M., Steininger, D., Fernandez Dominguez, G., “Wilddash - creating hazard-aware benchmarks”, in ECCV, 2018.

[118] Cour, T., Sapp, B., Taskar, B., “Learning from partial labels”, The Journal of Machine Learning Research, Vol. 12, 2011, pp. 1501–1536.

[119] Richter, S. R., Hayder, Z., Koltun, V., “Playing for benchmarks”, in ICCV, 2017, pp. 2232–2241.

[120] Cheng, B., Collins, M. D., Zhu, Y., Liu, T., Huang, T. S., Adam, H., Chen, L.-C., “Panoptic-deeplab: A simple, strong, and fast baseline for bottom-up panoptic segmentation”, in Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, 2020, pp. 12 475–12 485.

[121] Chollet, F. ç. o., “Xception: Deep learning with depthwise separable convolutions”, in CVPR, 2017.

[122] Wang, J., Sun, K., Cheng, T., Jiang, B., Deng, C., Zhao, Y., Liu, D., Mu, Y., Tan, M., Wang, X. *et al.*, “Deep high-resolution representation learning for visual recognition”, IEEE transactions on pattern analysis and machine intelligence, 2020.

[123] Šarić, J., Oršić, M., Antunović, T., Vražić, S., Šegvić, S., “Warp to the future: Joint forecasting of features and feature motion”, in Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, 2020, pp. 10 648–10 657.

[124] Šarić, J., Oršić, M., Antunović, T., Vražić, S., Šegvić, S., “Single level feature-to-feature forecasting with deformable convolutions”, in German Conference on Pattern Recognition. Springer, 2019, pp. 189–202.

[125] Kačan, M., Oršić, M., Šegvić, S., Ševrović, M., “Multi-task learning for irap attribute classification and road safety assessment”, in 2020 IEEE 23rd International Conference on Intelligent Transportation Systems (ITSC). IEEE, 2020, pp. 1–6.

[126] Grubišić, i., Oršić, M., Šegvić, S., “A baseline for semi-supervised learning of efficient semantic segmentation models”, in 17th International Conference on Machine Vision Applications, MVA 2021, Tokyo, Japan, July 25-27, 2021. IEEE, 2021.

[127] Bićanić, B., Oršić, M., Marković, I., Šegvić, S., Petrović, I., “Pedestrian tracking by probabilistic data association and correspondence embeddings”, in 2019 22th International Conference on Information Fusion (FUSION). IEEE, 2019, pp. 1–6.

[128] Oršić, M., Bevandić, P., Grubišić, I., Šarić, J., Šegvić, S., “Multi-domain semantic segmentation with pyramidal fusion”, arXiv preprint arXiv:2009.01636, 2020.

[129] Krešo, I., Oršić, M., Bevandić, P., Šegvić, S., "Robust semantic segmentation with ladder-densenet models", arXiv preprint arXiv:1806.03465, 2018.

[130] Bevandić, P., Oršić, M., Grubišić, I., Šarić, J., Šegvić, S., "Multi-domain semantic segmentation on datasets with overlapping classes".

[131] Bevandić, P., Krešo, I., Oršić, M., Šegvić, S., "Dense outlier detection and open-set recognition based on training with noisy negative images", arXiv preprint arXiv:2101.09193, 2021.

List of Figures

List of Tables

Biography

Marin Oršić was born in 1993 in Zagreb. He obtained his bachelor's and master's degrees at University of Zagreb, Faculty of Electrical Engineering and Computing. Upon finishing his master's thesis, he was employed at UniZG-FER in 2017. as part of the natural image understanding group conducted by professor Siniša Šegvić.

The first project he was involved in was "SafeTram: System for increased driving safety in public urban rail traffic", where he worked on real-time visual recognition. In 2019 he worked on "Development of a multi-functional anti-terrorism system (MAS)" project. His final project at FER, funded by Microblink Ltd., involved understanding of scene geometry using monocular cameras. He also worked as a consultant for Romb Technologies where he was involved in creating a data annotation workflow and an efficient computer vision inference pipeline for visual understanding of indoor warehouse scenery. During his engagement in the research lab of professor Šegvić, the group participated in two instances of the prestigious competition Robust Vision Challenge. In 2018 the group ranked second. In 2020 they won the competition while the submission was mainly based on the methodology from Marin's doctoral thesis. Currently, Marin is employed by Microblink Ltd. where he focuses on efficient computer vision applications.

His research interests include efficient visual recognition systems, dense prediction, semi-supervised learning and dense reconstruction algorithms. His other professional interests include efficient implementations for training models based on differentiable programming, object-oriented and data driven programming. He is involved in peer-reviewing for international journals and conferences.

List of Publications

Journal Papers

1. **Oršić, M., Šegvić, S., "Efficient semantic segmentation with pyramidal fusion", Pattern Recognition, 2020, p. 107611**

Conference Papers

1. **Oršić, M., Krešo, I., Bevandić, P., Šegvić, S., "In defense of pre-trained ImageNet architectures for real-time semantic segmentation of road-driving images", in Proceedings of the IEEE conference on computer vision and pattern recognition, 2019, pp. 12 607–12 616**
2. Šarić, J., Oršić, M., Antunović, T., Vražić, S., Šegvić, S., "Warp to the future: Joint forecasting of features and feature motion", in Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, 2020, pp. 10 648–10 657
3. Šarić, J., Oršić, M., Antunović, T., Vražić, S., Šegvić, S., "Single level feature-to-feature forecasting with deformable convolutions", in German Conference on Pattern Recognition. Springer, 2019, pp. 189–202
4. Bevandić, P., Krešo, I., Oršić, M., Šegvić, S., "Simultaneous semantic segmentation and outlier detection in presence of domain shift", in German Conference on Pattern Recognition. Springer, 2019, pp. 33–47
5. Kačan, M., Oršić, M., Šegvić, S., Ševrović, M., "Multi-task learning for irap attribute classification and road safety assessment", in 2020 IEEE 23rd International Conference on Intelligent Transportation Systems (ITSC). IEEE, 2020, pp. 1–6
6. Grubišić, i., Oršić, M., Šegvić, S., "A baseline for semi-supervised learning of efficient semantic segmentation models", in 17th International Conference on Machine Vision Applications, MVA 2021, Tokyo, Japan, July 25-27, 2021. IEEE, 2021
7. Bićanić, B., Oršić, M., Marković, I., Šegvić, S., Petrović, I., "Pedestrian tracking by probabilistic data association and correspondence embeddings", in 2019 22th International Conference on Information Fusion (FUSION). IEEE, 2019, pp. 1–6

Other Manuscripts

1. Oršić, M., Bevandić, P., Grubišić, I., Šarić, J., Šegvić, S., "Multi-domain semantic segmentation with pyramidal fusion", arXiv preprint arXiv:2009.01636, 2020
2. Bevandić, P., Krešo, I., Oršić, M., Šegvić, S., "Discriminative out-of-distribution detection for semantic segmentation", arXiv preprint arXiv:1808.07703, 2018

3. Krešo, I., Oršić, M., Bevandić, P., Šegvić, S., “Robust semantic segmentation with ladder-densenet models”, arXiv preprint arXiv:1806.03465, 2018
4. Bevandić, P., Oršić, M., Grubišić, I., Šarić, J., Šegvić, S., “Multi-domain semantic segmentation on datasets with overlapping classes”
5. Bevandić, P., Krešo, I., Oršić, M., Šegvić, S., “Dense outlier detection and open-set recognition based on training with noisy negative images”, arXiv preprint arXiv:2101.09193, 2021

Životopis

Marin Oršić je rođen 1993. u Zagrebu. Preddiplomski i diplomski studij završio je na Sveučilištu u Zagrebu, Fakultetu Elektrotehnike i Računarstva. Pri završetku diplosmkog studija 2017. zaposlio se na FER-u kao član grupe za razumjevanje prirodnih slika pod vodstvom profesora Siniše Šegvića.

Prvi projekt na kojem se angažirao bio je "SafeTram: Sustav za povećanje sigurnosti vožnje javnog urbanog tračničkog prometa", gdje je radio na vizualnom raspoznavanju u stvarnom vremenu. 2019. radio je na projektu "Razvoj multifunkcionalnog antiterorističkog sustava (MAS)". Na posljednjem projektu pod pokroviteljstvom tvrtke Microblink bavio se razumijevanjem geometrije scene korištenjem jednookih kamera. Radio je i kao konzultant za tvrtku Romb Technologies gdje se uključio u stvaranje procesa za označavanje podataka te cjevovoda za efikasnu izvedbu vizualnog razumijevanja scena iz skladišta. Tijekom rada u istraživačkom laboratoriju profesora Šegvića s grupom je sudjelovao na dvije inačice prestižnog natjecanja Robust Vision Challenge. 2018. je grupa ostvarila drugo mjesto. 2020. su pobijedili na natjecanju, a pobjednički podnesak se velikim djelom oslanja na metodologiju iz Marinove disertacije. Marin je trenutno zaposlen u tvrtki Microblink gdje se bavi efikasnim primjenama računalnog vida.

Njegova istraživačka područja uključuju efikasne sustave za vizualno raspoznavanje, gustu predikciju, polunadzirano učenje i algoritme za gustu rekonstrukciju. Njegovi ostali profesionalni interesi efikasne implementacije za treniranje modela temeljenih na diferenciabilnom programiranju, objektno orijentirano programiranje te programiranje vođeno podacima. Uključen je kao recenzent za međunarodne časopise i konferencije.