



Leibniz Transactions on  
**Embedded Systems**

**Volume 8 | Issue 1 | November 2022**  
**Special Issue on Embedded Systems for Computer Vision**



## ISSN 2199-2002

### *Published online and open access by*

the European Design and Automation Association (EDAA) / EMbedded Systems Special Interest Group (EMSIG) and Schloss Dagstuhl – Leibniz-Zentrum für Informatik GmbH, Dagstuhl Publishing, Saarbrücken/Wadern, Germany.

Online available at

<https://www.dagstuhl.de/dagpub/2199-2002>.

### *Publication date*

November 2022

### *Bibliographic information published by the Deutsche Nationalbibliothek*

The Deutsche Nationalbibliothek lists this publication in the Deutsche Nationalbibliografie; detailed bibliographic data are available in the Internet at <https://dnb.d-nb.de>.

### *License*

This work is licensed under a Creative Commons Attribution 4.0 International license (CC BY 4.0): <http://creativecommons.org/licenses/by/4.0>



In brief, this license authorizes each and everybody to share (to copy, distribute and transmit) the work under the following conditions, without impairing or restricting the authors' moral rights:

- Attribution: The work must be attributed to its authors.

The copyright is retained by the corresponding authors.

### *Digital Object Identifier*

10.4230/LITES-v008-i001

### *Aims and Scope*

LITES aims at the publication of high-quality scholarly articles, ensuring efficient submission, reviewing, and publishing procedures. All articles are published open access, i.e., accessible online without any costs. The rights are retained by the author(s).

LITES publishes original articles on all aspects of embedded computer systems, in particular: the design, the implementation, the verification, and the testing of embedded hardware and software systems; the theoretical foundations; single-core, multi-processor, and networked architectures and their energy consumption and predictability properties; reliability and fault tolerance; security properties; and on applications in the avionics, the automotive, the telecommunication, the medical, and the production domains.

### *Editorial Board*

- Alan Burns (Editor-in-Chief)
- Bashir Al Hashimi
- Karl-Erik Arzen
- Neil Audsley
- Sanjoy Baruah
- Samarjit Chakraborty
- Marco di Natale
- Martin Fränzle
- Steve Goddard
- Gernot Heiser
- Axel Jantsch
- Sang Lyul Min
- Lothar Thiele
- Virginie Wiels

### *Editorial Office*

Michael Wagner (*Managing Editor*)

Michael Didas (*Managing Editor*)

Jutka Gasiorowski (*Editorial Assistance*)

Dagmar Glaser (*Editorial Assistance*)

Thomas Schillo (*Technical Assistance*)

### *Contact*

Schloss Dagstuhl – Leibniz-Zentrum für Informatik

LITES, Editorial Office

Oktavie-Allee, 66687 Wadern, Germany

[lites@dagstuhl.de](mailto:lites@dagstuhl.de)

<http://www.dagstuhl.de/lites>





## ■ Contents

Introduction to the Special Issue on Embedded Systems for Computer Vision <i>Samarjit Chakraborty and Qing Rao</i> .....	0:1–0:8
---	---------

### Papers

Susceptibility to Image Resolution in Face Recognition and Training Strategies to Enhance Robustness <i>Martin Knoche, Stefan Hörmann, and Gerhard Rigoll</i> .....	1:1–1:20
Micro- and Macroscopic Road Traffic Analysis using Drone Image Data <i>Friedrich Kruber, Eduardo Sánchez Morales, Robin Egolf, Jonas Wurst, Samarjit Chakraborty, and Michael Botsch</i> .....	2:1–2:27
HW-Flow: A Multi-Abstraction Level HW-CNN Codesign Pruning Methodology <i>Manoj-Rohit Vemparala, Nael Fasfous, Alexander Frickenstein, Emanuele Valpreda, Manfredi Camalleri, Qi Zhao, Christian Unger, Naveen-Shankar Nagaraja, Maurizio Martina, and Walter Stechele</i> .....	3:1–3:30





# Introduction to the Special Issue on Embedded Systems for Computer Vision

Samarjit Chakraborty   

The University of North Carolina (UNC) at Chapel Hill, US

Qing Rao 

Momenta Europe GmbH, Stuttgart, Germany

We provide a broad overview of some of the current research directions at the intersection of embedded systems and computer vision, in addition to introducing the papers appearing in this special issue. Work at this intersection is steadily growing in importance, especially in the context of autonomous and cyber-physical systems design. Vision-based perception is almost a mandatory component in any autonomous system, but also adds myriad challenges like, how to efficiently implement vision

processing algorithms on resource-constrained embedded architectures, and how to verify the functional and timing correctness of these algorithms. Computer vision is also crucial in implementing various smart functionality like security, e.g., using facial recognition, or monitoring events or traffic patterns. Some of these applications are reviewed in this introductory article. The remaining articles featured in this special issue dive into more depth on a few of them.

**2012 ACM Subject Classification** Computer systems organization → Embedded and cyber-physical systems

**Keywords and Phrases** Embedded systems, Computer vision, Cyber-physical systems, Computer architecture

**Digital Object Identifier** 10.4230/LITES.8.1.0

**Funding** *Samarjit Chakraborty*: Acknowledges support from the US NSF grant# 2038960.

**Published** 2022-11-16

**Editor** Samarjit Chakraborty and Qing Rao

**Special Issue** Special Issue on Embedded Systems for Computer Vision

## 1 Computer vision and embedded systems

Efficiently implementing computer vision algorithms on resource-constrained embedded systems is necessary for many application domains and is continuing to attract widespread attention in the research community [50]. We are witnessing a surge in the development of various smart and autonomous systems – such as autonomous cars, robots, drones, and industrial automation systems. All of these systems rely on environmental perception in order to generate the necessary control action. Towards this, inputs from various sensors like cameras and lidars need to be processed and their output serves as inputs to control algorithms that compute commands for realizing the desired system functionality. Hence, vision processing algorithms have to meet various architectural and resource constraints and need to be certified for functional and timing correctness in order to ensure the reliability of the entire system. This has triggered research on the development of high-performance embedded systems architectures to support computer vision solutions, e.g., using accelerators like FPGAs and GPUs, in particular for those that rely on machine learning. There has also been recent work on how to provide timing guarantees to computer vision algorithms that are a part of feedback control loops. Power and memory optimization of computer vision algorithms, and issues related to privacy and security of vision-based applications and systems have also been published during the past couple of years.

In view of these developments, we organized this special issue for the Leibniz Transactions on Embedded Systems and invited papers on a variety of topics at this intersection of computer vision and embedded systems. The topics we listed included new embedded systems architectures



© S. Chakraborty and Q. Rao;  
licensed under Creative Commons Attribution 4.0 International (CC BY 4.0)

*Leibniz Transactions on Embedded Systems*, Vol. 8, Issue 1, Article No. 0, pp. 00:1–00:8



Leibniz Transactions on Embedded Systems

LITES Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

– including FPGAs, GPUs, and heterogeneous MpSoCs – for computer vision, novel algorithms for computer vision targeting embedded applications, machine learning and neural networks for image and video understanding for autonomous systems, timing analysis of computer vision algorithms and architectures, performance and power analysis and management of computer vision systems, vision-based control or visual servoing systems, security and privacy issues in vision-based embedded systems, robustness issues in vision-based autonomous systems, and debugging vision-based embedded systems. Papers on applications of computer vision were also invited.

Before we introduce the three papers that comprise this special issue, we briefly review some of the recent literature on the above topics. This review is by no means complete or detailed. Its only purpose is to provide a context for this special issue and impress upon the reader the variety of work that has been done in this area, and the rich set of challenges that are still remaining.

### 1.1 Control + Vision

As we already hinted above, almost all autonomous systems rely on different feedback controllers that need to be implemented on resource-constrained distributed embedded systems [5]. In the past, controller *design* and their *implementation* were done separately, which resulted in a mismatch between model-level assumptions and implementation realities. This necessitated an iterative design approach and significant effort in testing and debugging. More recently, different parts of a control system are being co-designed and co-synthesized [42].

In the case of *visual servoing* systems, where a computer vision system is used to compute control inputs, co-design is also being used. More specifically, while controller design was traditionally done independently and was disconnected from the vision or perception processing techniques, recently the need for co-design between the two is being recognized [19, 53]. Holistic approaches to closed-loop behavior have also been studied in [44], instead of designing components like object detection, tracking, motion planning, and multi-sensor fusion in isolation, followed by integrating these components together. By co-designing and optimizing these components together, it has been shown that resource utilization may be significantly reduced with minimal impact on performance such as motion planning.

Several studies have focused on evaluation of vision-in-the-loop systems, since building a full prototype of such systems might be too expensive and infeasible. Hence, the evaluation is done in a progressive manner as different components of the system are designed. Here, the work in [21] has proposed an evaluation framework for *model-in-the-loop*, *software-in-the-loop*, and *processor-in-the-loop* simulation features, as the design progresses from the model of the system, to the software and its deployment on multi-core processors. The goal of this work was to evaluate the closed-loop performance of industrial motion control systems. The design of the control strategy, and the design and implementation of the vision processing system – both involve several parameters. How to jointly determine the values of these parameters in order to optimize control performance, and also resource usage, is the main problem. Today, only “point solutions” exist and we will continue to see more work in this area.

### 1.2 Efficient implementation of vision processing (incl. GPUs & FPGAs)

Efficient implementation of compute-intensive tasks on resource-constrained architectures, such as those seen in the automotive domain, require the use of various forms of hardware accelerators like FPGAs [48]. Towards this, [20] proposed a FPGA-based scalable and resource-efficient multi-camera GigE Vision IP core for video and image processing. This IP core, that was implemented on a Xilinx Virtex-4 FPGA, supports the connection of multi-camera interfaces to a single Gigabit

Ethernet port using an Ethernet switch. Similar FPGA-based implementations have also been studied in [29] and [49]. Since modern cars are now equipped with multiple cameras, lidar, and radar sensors, the volume of data that needs to be acquired for efficient processing is also large and efficient data acquisition techniques and communication architectures have also been studied [14].

Sensor fusion is a standard task in many domains like automotive and robotics, and can be very resource-heavy. Therefore, techniques for efficient sensor fusion has attracted considerable attention. For example, [43] proposes techniques for spatiotemporal sampling to activate Lidars only at regions-of-interest that are identified by analyzing the visual input. Such a task-based sampling approach significantly reduces the volume of data that is sensed and transferred, thereby reducing sensor fusion workload. A similar approach, where a smart camera captures only task-critical information and is driven by embedded deep neural network algorithms for real-time control of sensor parameters has been presented in [32].

As outlined above, in most vision-based control applications, the vision processing algorithms incur very heavy computational workload. The work in [8] studied approximate image processing techniques to reduce this workload, making vision-based control suitable for embedded platforms. The underlying control strategy was adapted to an approximation-aware optimal linear-quadratic-gaussian (LQG) controller, where the approximation error was modeled as sensor noise. This work may also be viewed as a *co-design* between control strategy and its implementation, along the lines of other studies like [19, 53]. Further, the tradeoffs between the degree of approximation, and the resulting closed-loop quality-of-control, and metrics like memory utilization and energy consumption have been studied in [7].

Yet another example of efficient implementation of vision-based control is in [30], which proposes identifying different *system scenarios* that are optimized, and a switched controller switches between these scenarios. Identifying only a limited number of scenarios helps with optimization that would not be possible otherwise. Along similar lines, the work in [31] attempts to reduce sensor-to-actuator delay in vision-based control applications by pipelining the vision task on a multiprocessor architecture. In particular, this work shows how such pipelining may be implemented for model predictive control, while accounting for workload variations in the different stages of the image processing pipeline. As a continuation of this study, how the number of pipeline stages impacts the quality of control has been studied in [45].

### 1.3 Verification and monitoring

Safety [57] and security [54] of autonomous systems is usually of crucial importance. While there has been considerable work on the functional [17] and also timing verification [47] of cyber-physical systems, and in particular controllers implementing different autonomous features, a full verification of the system also requires a verification of the vision processing and understanding algorithms that provide inputs into the controllers. Towards this, monitoring [13], debugging [12] and optimization techniques for FPGA-based implementation of vision processing [11] have also attracted considerable attention recently. However, work in this area is relatively nascent; as systems become more complex and the need for certification increases, we will see more results in this domain.

### 1.4 Timing predictable vision processing

While mainstream computer vision has considered *fast* vision processing algorithms, the issue of *real-time*, viz., guaranteeing that the output is produced within a deadline, has not been studied. However, when used in conjunction with control algorithms and in safety-critical systems, such real-time processing guarantees become necessary. Towards this, temporal isolation might be

needed between soft real-time applications like computer vision and time-critical applications like control tasks [27]. Although many certification techniques rely on time partitioning, it has been shown that the use of multicore + accelerator platforms can disrupt such partitioning or timing isolation schemes [3]. In this context, how to design the vision processing system to be timing predictable [2, 4], and how to debug systems for timing violations have also been studied [41].

Most control systems tend to have a certain degree of timing robustness [18], and this can be used for the design of the perception processing subsystem. How the behavior of vision-in-the-loop control systems is impacted by the processing platform and the control software executing on it has been studied in [21]. This study, in particular, evaluates the *predictability* of the embedded computational platform “CompSOC” [16], which guarantees periodic and deterministic execution of control tasks, allowing a verification of their timing properties.

**Timing predictable GPU processing:** A number of papers have recently investigated the role of GPUs in providing such timing guarantees. It has been established that using OpenCV-based applications on GPUs, unexpected delays might occur not only on the GPUs, but also on the host CPUs, which might jeopardize the real-time constraints associated with vision-based applications [1]. While most studies have focused on how to use GPUs to gain computational advantage, there has been relatively less work done on evaluating the real-time characteristics of GPUs – from both AMD and also NVIDIA [34]. Allocation strategies for real-time management of multi-GPU systems has been studied in [9].

## 1.5 Algorithms and data structures for efficient vision processing

There exists a variety of algorithms, processing architectures, and techniques for implementing vision processing tasks on embedded computing platforms [28]. In particular, a lot of recent attention has been on neural network based processing and their implications, like ensuring consistency in their performance [51], or efficiently implementing them on resource-constrained edge devices [15]. To cite examples from specific domains – in-vehicle augmented reality applications [40] and autonomous features in cars (such as automated parking or driving) require new data structures and algorithmic techniques for vision processing [37]. More importantly, these have to be resource efficient, e.g., so that complex 3D shapes of the environment can be transmitted within the vehicle using low- to medium-bandwidth communication infrastructure [36]. In addition to suitable data structures, special compression techniques for resource efficient in-vehicle communication and computation have also been studied [38, 35].

The use of neural networks for image processing on embedded systems has been widely studied in the literature [52, 58], along with specialized hardware architectures for vision processing [26, 29]. When multiple convolutional neural networks (CNNs) with each processing a different video stream are implemented on the same resource-constrained embedded platform, then their inference and timing performance might not be satisfactory. However, reorganizing these CNNs and exploiting parallelism, pipelining, and merging the images, might lead to significant improvements [56]. Some studies have also considered different graph transformation [10] and scheduling strategies for OpenVX graphs to achieve better real-time performance [55]. OpenVX is a standard for cross-platform acceleration of computer vision algorithms, and provides a high-level of abstraction for programming vision applications [33].

In many cost-sensitive domains like automotive and also for general purpose cost-effective imaging, a common challenge is to solve problems using low-cost and resource-constrained components. For example high dynamic range images may be recovered by fusing multiple low dynamic range (LDR) images. If all the LDR images are perfectly aligned, then this fusion process is much easier. However, in reality, there are misalignments and jitters between different

cameras and several studies have proposed machine learning techniques for correcting them (e.g., see [25]). This is especially relevant in the automotive and robotics domains where physical movement causes cameras to vibrate. Another example in the same direction – where intelligent algorithmic or machine learning techniques are used to utilize low-cost or resource-constrained components – is where 3D shapes of objects are created from a single image using deep neural networks in the context of autonomous driving [39].

As outlined above, there has been a tremendous growth in the use of machine learning algorithms for vision processing in various embedded computing settings, especially in the automotive domain. But a big challenge in their use is the need for a large amount of labelled training data, especially because often the data needs to be manually labeled. To address this, various techniques have been proposed to reduce the volume of such data that is needed. One such technique is referred to as *active learning*, where a model selects samples for labeling based on their uncertainty, thereby reducing the volume of data needed for various tasks like 3D object detection [46].

## 1.6 Other applications

While we have so far mostly focused on autonomous systems and their instantiations in domains like automotive and robotics, vision processing is also useful in various monitoring applications for example for monitoring traffic flow [23] to devise suitable traffic analysis [24] and management strategies [6]. Embedded vision processing has also been used for security and face recognition, where various efficient implementation techniques have been studied, including how to best use combinations of cloud and edge processing [22].

## 2 Papers in this special issue

This special issue features three papers. The first, entitled “*Susceptibility to Image Resolution in Face Recognition and Training Strategies to Enhance Robustness*” by Knoche, Hörmann and Rigoll studies how facial recognition algorithms are susceptible to the resolution of the input images. They show that recognition accuracy can dramatically drop when the image resolution is reduced. Since training images and input images that need to be recognized might not have the same resolution, this finding poses a serious problem. To address this, they have proposed new training strategies on state-of-the-art face recognition models, which provided significant boost in accuracy and improved robustness.

The second paper, entitled “*Micro- and Macroscopic Road Traffic Analysis using Drone Image Data*” by Kruber *et al.* discusses analysis of traffic image data that is captured using drones. This includes estimating vehicle states and trajectories, and also macroscopic statistics such as traffic flow and traffic density. Finally, the third paper, “*HW-Flow: A Multi-Abstraction Level HW-CNN Codesign Pruning Methodology*” by Vemparala *et al.* is on optimizing convolutional neural network models, that are widely used for image classification, segmentation, and object detection tasks in autonomous vehicles. Towards this, the paper proposes a framework referred to as *HW-Flow* that achieved around 2x reduction in energy and latency in a number of well-known neural networks and datasets.

We hope that readers will find these articles to be interesting and will gain new insights into this evolving area of embedded systems for computer vision. We thank everyone who submitted their research to this special issue. We also thank all the reviewers, the EiC – Alan Burns – and members of the LITES Editorial Office, especially Michael Wagner, without whose help this special issue would not have been possible.

## References

- 1 Tanya Amert and James H. Anderson. Cupid<sup>tt</sup>: Detecting improper GPU usage in real-time applications. In *24th IEEE International Symposium on Real-Time Distributed Computing (ISORC)*, 2021.
- 2 Tanya Amert, Michael Balszun, Martin Geier, F. Donelson Smith, James H. Anderson, and Samarjit Chakraborty. Timing-predictable vision processing for autonomous systems. In *Design, Automation & Test in Europe Conference & Exhibition (DATE)*, 2021.
- 3 Tanya Amert, Zelin Tong, Sergey Voronov, Joshua Bakita, F. Donelson Smith, and James H. Anderson. Timewall: Enabling time partitioning for real-time multicore+accelerator platforms. In *42nd IEEE Real-Time Systems Symposium (RTSS)*, 2021.
- 4 Michael Balszun, Martin Geier, and Samarjit Chakraborty. Predictable vision for autonomous systems. In *23rd IEEE International Symposium on Real-Time Distributed Computing (ISORC)*, 2020.
- 5 Wanli Chang and Samarjit Chakraborty. Resource-aware automotive control systems design: A cyber-physical systems approach. *Found. Trends Electron. Des. Autom.*, 10(4):249–369, 2016.
- 6 Wanli Chang, Debayan Roy, Shuai Zhao, Anuradha Annaswamy, and Samarjit Chakraborty. Cps-oriented modeling and control of traffic signals using adaptive back pressure. In *Design, Automation & Test in Europe Conference & Exhibition (DATE)*, 2020.
- 7 Sayandip De, Sajid Mohamed, Konstantinos Bimpisidis, Dip Goswami, Twan Basten, and Henk Corporaal. Approximation trade offs in an image-based control system. In *Design, Automation & Test in Europe Conference & Exhibition (DATE)*, 2020.
- 8 Sayandip De, Sajid Mohamed, Dip Goswami, and Henk Corporaal. Approximation-aware design of an image-based control system. *IEEE Access*, 8:174568–174586, 2020.
- 9 Glenn A. Elliott, Bryan C. Ward, and James H. Anderson. GPUSync: A framework for real-time GPU management. In *34th IEEE Real-Time Systems Symposium (RTSS)*, 2013.
- 10 Glenn A. Elliott, Kecheng Yang, and James H. Anderson. Supporting real-time computer vision workloads using OpenVX on Multicore+GPU platforms. In *IEEE Real-Time Systems Symposium (RTSS)*, 2015.
- 11 Martin Geier, Marian Brändle, and Samarjit Chakraborty. Insert & save: Energy optimization in IP core integration for FPGA-based real-time systems. In *27th IEEE Real-Time and Embedded Technology and Applications Symposium (RTAS)*, 2021.
- 12 Martin Geier, Marian Brändle, Dominik Faller, and Samarjit Chakraborty. Debugging FPGA-accelerated real-time systems. In *IEEE Real-Time and Embedded Technology and Applications Symposium (RTAS)*, 2020.
- 13 Martin Geier, Dominik Faller, Marian Brändle, and Samarjit Chakraborty. Cost-effective energy monitoring of a Zynq-based real-time system including dual Gigabit Ethernet. In *27th IEEE Annual International Symposium on Field-Programmable Custom Computing Machines (FCCM)*, 2019.
- 14 Martin Geier, Florian Pitzl, and Samarjit Chakraborty. GigE vision data acquisition for visual servoing using SG/DMA proxying. In *14th ACM/IEEE Symposium on Embedded Systems for Real-Time Multimedia (ESTIMedia)*, 2016.
- 15 Abhinav Goel, Caleb Tung, Xiao Hu, George K. Thiruvathukal, James C. Davis, and Yung-Hsiang Lu. Efficient computer vision on edge devices with pipeline-parallel hierarchical neural networks. In *27th Asia and South Pacific Design Automation Conference (ASP-DAC)*, 2022.
- 16 Kees Goossens, Arnaldo Azevedo, Karthik Chandrasekar, Manil Dev Gomony, Sven Goossens, Martijn Koedam, Yonghui Li, Davit Mirzoyan, Anca Mariana Molnos, Ashkan Beyrandvand Nejad, Andrew Nelson, and Shubhendu Sinha. Virtual execution platforms for mixed-time-criticality systems: the compsoc architecture and design flow. *SIGBED Rev.*, 10(3):23–34, 2013.
- 17 Dip Goswami, Reinhard Schneider, and Samarjit Chakraborty. Re-engineering cyber-physical control applications for hybrid communication protocols. In *Design, Automation and Test in Europe (DATE)*, 2011.
- 18 Dip Goswami, Reinhard Schneider, and Samarjit Chakraborty. Relaxing signal delay constraints in distributed embedded controllers. *IEEE Trans. Control. Syst. Technol.*, 22(6):2337–2345, 2014.
- 19 Clara Hobbs, Debayan Roy, Parasara Sridhar Dugirala, F. Donelson Smith, Soheil Samii, James H. Anderson, and Samarjit Chakraborty. Perception computing-aware controller synthesis for autonomous systems. In *Design, Automation & Test in Europe Conference & Exhibition (DATE)*, 2021.
- 20 Omar W Ibraheem, Arif Irwansyah, Jens Hagemeyer, Mario Porrmann, and Ulrich Rueckert. A resource-efficient multi-camera GiGE vision IP core for embedded vision processing platforms. In *IEEE International Conference on ReConfigurable Computing and FPGAs (ReConFig)*, 2015.
- 21 Chaitanya Jugade, Daniel Hartgers, Phan Dúc Anh, Sajid Mohamed, Mojtaba Haghi, Dip Goswami, Andrew Nelson, Gijs van der Veen, and Kees Goossens. An evaluation framework for vision-in-the-loop motion control systems. In *27th IEEE International Conference on Emerging Technologies and Factory Automation (ETFA)*, 2022.
- 22 Anis Koubaa, Adel Ammar, Anas Kanhouch, and Yasser AlHabashi. Cloud versus edge deployment strategies of real-time face recognition inference. *IEEE Transactions on Network Science and Engineering*, 9(1):143–160, 2021.
- 23 Friedrich Kruber, Eduardo Sánchez Morales, Samarjit Chakraborty, and Michael Botsch. Vehicle position estimation with aerial imagery from unmanned aerial vehicles. In *IEEE Intelligent Vehicles Symposium (IV)*, 2020.



- 24 Friedrich Kruber, Jonas Wurst, Eduardo Sánchez Morales, Samarjit Chakraborty, and Michael Botsch. Unsupervised and supervised learning with the random forest algorithm for traffic scenario clustering and classification. In *IEEE Intelligent Vehicles Symposium (IV)*, 2019.
- 25 Zhen Liu, Wenjie Lin, Xinpeng Li, Qing Rao, Ting Jiang, Mingyan Han, Haoqiang Fan, Jian Sun, and Shuaicheng Liu. Adnet: Attention-guided deformable convolutional network for high dynamic range imaging. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR) Workshops*, 2021.
- 26 Dipan Kumar Mandal, Jagadeesh Sankaran, Akshay Gupta, Kyle Castille, Shraddha Gondkar, Sanmati Kamath, Pooja Sundar, and Alan Phipps. An embedded vision engine (EVE) for automotive vision processing. In *IEEE International Symposium on Circuits and Systems (ISCAS)*, 2014.
- 27 Alejandro Masrur, Sebastian Drössler, Thomas Pfeuffer, and Samarjit Chakraborty. VM-based real-time services for automotive control applications. In *16th IEEE International Conference on Embedded and Real-Time Computing Systems and Applications (RTCSA)*, 2010.
- 28 Mahmoud Méribout, Asma Baobaid, Mohamed Ould-Khaoua, Varun Kumar Tiwari, and Juan Pablo Pena. State of art iot and edge embedded systems for real-time machine vision applications. *IEEE Access*, 10:58287–58301, 2022.
- 29 Seifeddine Messaoud, Soulef Bouaafia, Amna Maraoui, Ahmed Chiheb Ammari, Lazhar Khriji, and Mohsen Machhout. Deep convolutional neural networks-based hardware-software on-chip system for computer vision application. *Comput. Electr. Eng.*, 98:107671, 2022.
- 30 Sajid Mohamed, Dip Goswami, Vishak Nathan, Raghu Rajappa, and Twan Basten. A scenario- and platform-aware design flow for image-based control systems. *Microprocess. Microsystems*, 75:103037, 2020.
- 31 Sajid Mohamed, Nilay Saraf, Daniele Bernardini, Dip Goswami, Twan Basten, and Alberto Bemporad. Adaptive predictive control for pipelined multiprocessor image-based control systems considering workload variations. In *59th IEEE Conference on Decision and Control (CDC)*, 2020.
- 32 Burhan Ahmad Mudassar, Priyabrata Saha, Yun Long, Mohammad Faisal Amir, Evan Gebhardt, Taesik Na, Jong Hwan Ko, Marilyn Wolf, and Saibal Mukhopadhyay. CAMEL: an adaptive camera with embedded machine learning-based sensor parameter control. *IEEE J. Emerg. Sel. Topics Circuits Syst.*, 9(3):498–508, 2019.
- 33 The Khronos Group, OpenVX: Portable, power efficient vision processing. <https://www.khronos.org/openvx/>.
- 34 Nathan Otterness and James H. Anderson. AMD GPUs as an alternative to NVIDIA for supporting real-time workloads. In *32nd Euromicro Conference on Real-Time Systems (ECRTS)*, volume 165 of *LIPICs*, pages 10:1–10:23. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2020.
- 35 Qing Rao and Samarjit Chakraborty. Efficient lossless compression for depth information in traffic scenarios. *Multim. Syst.*, 25(4):293–306, 2019.
- 36 Qing Rao and Samarjit Chakraborty. In-vehicle object-level 3D reconstruction of traffic scenes. *IEEE Trans. Intell. Transp. Syst.*, 22(12):7747–7759, 2021.
- 37 Qing Rao, Christian Grünler, Markus Hammori, and Samarjit Chakraborty. Design methods for augmented reality in-vehicle infotainment systems. In *51st Annual Design Automation Conference (DAC)*, 2014.
- 38 Qing Rao, Christian Grünler, Markus Hammori, and Samarjit Chakraborty. Stixel on the bus: An efficient lossless compression scheme for depth information in traffic scenarios. In *20th Anniversary International Conference of MultiMedia Modeling (MMM)*, volume 8325 of *Lecture Notes in Computer Science*. Springer, 2014.
- 39 Qing Rao, Lars Krüger, and Klaus Dietmayer. Monocular 3D shape reconstruction using deep neural networks. In *IEEE Intelligent Vehicles Symposium (IV)*, 2016.
- 40 Qing Rao, Tobias Tropper, Christian Grünler, Markus Hammori, and Samarjit Chakraborty. AR-IVI - implementation of in-vehicle augmented reality. In *IEEE International Symposium on Mixed and Augmented Reality (ISMAR)*, 2014.
- 41 Debayan Roy, Clara Hobbs, James H. Anderson, Marco Caccamo, and Samarjit Chakraborty. Timing debugging for cyber-physical systems. In *Design, Automation & Test in Europe Conference & Exhibition (DATE)*, 2021.
- 42 Debayan Roy, Licong Zhang, Wanli Chang, Sanjoy K. Mitter, and Samarjit Chakraborty. Semantics-preserving cosynthesis of cyber-physical systems. *Proc. IEEE*, 106(1):171–200, 2018.
- 43 Kruttidipta Samal, Hemant Kumawat, Priyabrata Saha, Marilyn Wolf, and Saibal Mukhopadhyay. Task-driven rgb-lidar fusion for object tracking in resource-efficient autonomous system. *IEEE Trans. Intell. Veh.*, 7(1):102–112, 2022.
- 44 Kruttidipta Samal, Marilyn Wolf, and Saibal Mukhopadhyay. Closed-loop approach to perception in autonomous system. In *Design, Automation & Test in Europe Conference & Exhibition (DATE)*, 2021.
- 45 Róbinson Medina Sánchez, Juan Valencia, Sander Stuijk, Dip Goswami, and Twan Basten. Designing a controller with image-based pipelined sensing and additive uncertainties. *ACM Trans. Cyber Phys. Syst.*, 3(3):33:1–33:26, 2019.
- 46 Sebastian Schmidt, Qing Rao, Julian Tatsch, and Alois C. Knoll. Advanced active learning strategies for object detection. In *IEEE Intelligent Vehicles Symposium (IV)*, 2020.
- 47 Reinhard Schneider, Dip Goswami, Alejandro Masrur, Martin Becker, and Samarjit Chakraborty. Multi-layered scheduling of mixed-criticality cyber-physical systems. *J. Syst. Archit.*, 59(10-D):1215–1230, 2013.
- 48 Shreejith Shanker, Philipp Mundhenk, Andreas Ettner, Suhaib A. Fahmy, Sebastian Steinhorst, Martin Lukaszewycz, and Samarjit Chakraborty. VEGA: A high performance vehicular ethernet gateway on hybrid FPGA. *IEEE Trans. Computers*, 66(10):1790–1803, 2017.
- 49 Ponnann Suresh, Saravanakumar Umathurai, Celestine Iwendi, Senthilkumar Mohan, and Gautam

- Srivastava. Field-programmable gate arrays in a low power vision system. *Comput. Electr. Eng.*, 90:106996, 2021.
- 50 George K. Thiruvathukal and Yung-Hsiang Lu. Efficient computer vision for embedded systems. *Computer*, 55(4):15–19, 2022.
- 51 Caleb Tung, Abhinav Goel, Fischer Bordwell, Nick Eliopoulos, Xiao Hu, Yung-Hsiang Lu, and George K. Thiruvathukal. Why accuracy is not enough: The need for consistency in object detection. *IEEE Multim.*, 29(3):8–16, 2022.
- 52 R. Udendhran, M. Balamurugan, Annamalai Suresh, and R. Varatharajan. Enhancing image processing architecture using deep learning for embedded vision systems. *Microprocess. Microsystems*, 76, 2020.
- 53 Zhilu Wang, Chao Huang, Yixuan Wang, Clara Hobbs, Samarjit Chakraborty, and Qi Zhu. Bounding perception neural network uncertainty for safe control of autonomous systems. In *Design, Automation & Test in Europe Conference & Exhibition (DATE)*, 2021.
- 54 Peter Waszecki, Philipp Mundhenk, Sebastian Steinhorst, Martin Lukasiewicz, Ramesh Karri, and Samarjit Chakraborty. Automotive electrical and electronic architecture security via distributed in-vehicle traffic monitoring. *IEEE Trans. Comput. Aided Des. Integr. Circuits Syst.*, 36(11):1790–1803, 2017.
- 55 Ming Yang, Tanya Amert, Kecheng Yang, Nathan Otterness, James H. Anderson, F. Donelson Smith, and Shige Wang. Making openvx really "real time". In *IEEE Real-Time Systems Symposium (RTSS)*, 2018.
- 56 Ming Yang, Shige Wang, Joshua Bakita, Thanh Vu, F. Donelson Smith, James H. Anderson, and Jan-Michael Frahm. Re-thinking CNN frameworks for time-sensitive autonomous-driving applications: Addressing an industrial challenge. In *25th IEEE Real-Time and Embedded Technology and Applications Symposium (RTAS)*, 2019.
- 57 Anand Yeolekar, Ravindra Metta, Clara Hobbs, and Samarjit Chakraborty. Checking scheduling-induced violations of control safety properties. In *20th International Symposium on Automated Technology for Verification and Analysis (ATVA)*, volume 13505 of *Lecture Notes in Computer Science*. Springer, 2022.
- 58 Junxing Zhang, Shuo Yang, Chunjuan Bo, and Zhiyuan Zhang. Vehicle logo detection based on deep convolutional networks. *Comput. Electr. Eng.*, 90:107004, 2021.

# Susceptibility to Image Resolution in Face Recognition and Training Strategies to Enhance Robustness

Martin Knoche ✉ 

Technische Universität München, Arcisstrasse 21 80333 München, Deutschland

Stefan Hörmann ✉

Technische Universität München, Arcisstrasse 21 80333 München, Deutschland

Gerhard Rigoll ✉ 

Technische Universität München, Arcisstrasse 21 80333 München, Deutschland

## Abstract

Many face recognition approaches expect the input images to have similar image resolution. However, in real-world applications, the image resolution varies due to different image capture mechanisms or sources, affecting the performance of face recognition systems. This work first analyzes the image resolution susceptibility of modern face recognition. Face verification on the very popular LFW dataset drops from 99.23% accuracy to almost 55% when image dimensions of both images are reduced to arguable very poor resolution. With cross-resolution image pairs (one HR and one LR image), face verification accuracy is even worse. This characteristic is investigated more in-depth by analyzing the feature distances utilized for face verification. To increase the robustness, we propose two training strategies applied to a state-of-the-art face recognition model: 1) Training with 50% low resolution images within each batch and 2) using the cosine distance loss between high and low resolution features in a si-

mese network structure. Both methods significantly boost face verification accuracy for matching training and testing image resolutions. Training a network with different resolutions simultaneously instead of adding only one specific low resolution showed improvements across all resolutions and made a single model applicable to unknown resolutions. However, models trained for one particular low resolution perform better when using the exact resolution for testing. We improve the face verification accuracy from 96.86% to 97.72% on the popular LFW database with uniformly distributed image dimensions between  $112 \times 112$ px and  $5 \times 5$ px. Our approaches improve face verification accuracy even more from 77.56% to 87.17% for distributions focusing on lower images resolutions. Lastly, we propose specific image dimension sets focusing on high, mid, and low resolution for five well-known datasets to benchmark face verification accuracy in cross-resolution scenarios.

**2012 ACM Subject Classification** Computing methodologies → Neural networks

**Keywords and Phrases** recognition, resolution, cross, face, identification

**Digital Object Identifier** 10.4230/LITES.8.1.1

**Supplementary Material** *Dataset (Evaluation Protocols)*: <https://github.com/martlgap/btm-stm>

**Funding** This work was partially supported by “Bayerische Staatsministerium für Wirtschaft, Energie und Technologie” within the framework of a funding program of “Informations- und Kommunikationstechnik” for the project “Grundrechtskonforme Gesichtserkennung im öffentlichen Raum” (e-freedom).

**Received** 2020-12-15 **Accepted** 2022-01-25 **Published** 2022-11-16

**Editor** Samarjit Chakraborty and Qing Rao

**Special Issue** Special Issue on Embedded Systems for Computer Vision

## 1 Introduction

Over the last few years, face recognition has gained progressively more attraction. Szegedy et al. [24] introduced one of the first deep-learning-based approaches in 2014 and applied a 9-layer convolutional neural network. Since then, deep-learning-based approaches have evolved more and more due to the growing availability of powerful GPUs and novel large datasets, e.g., Microsoft’s

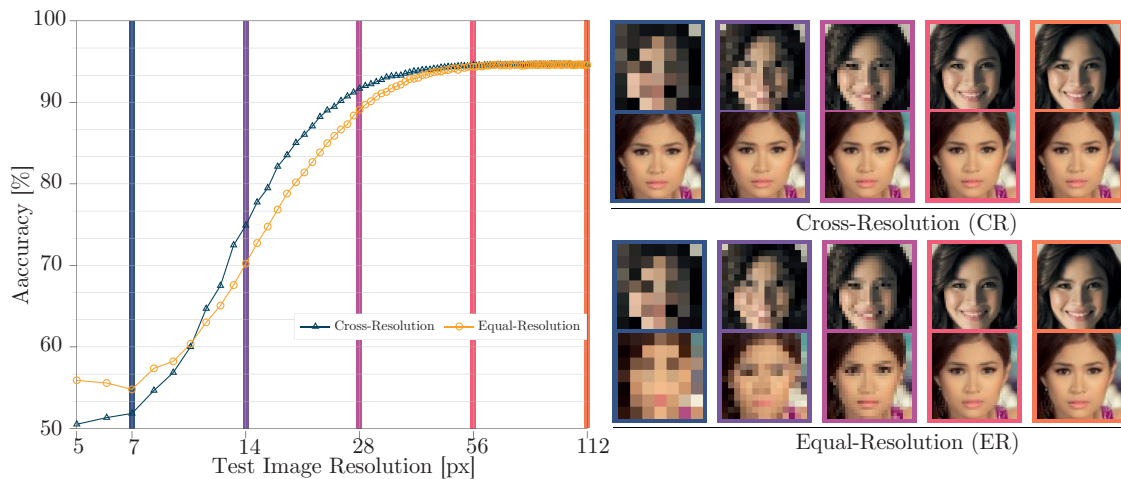


© Martin Knoche, Stefan Hörmann, and Gerhard Rigoll;  
licensed under Creative Commons Attribution 4.0 International (CC BY 4.0)  
*Leibniz Transactions on Embedded Systems*, Vol. 8, Issue 1, Article No. 1, pp. 01:1–01:20



Leibniz Transactions on Embedded Systems  
LITES Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

## 01:2 Image Resolution Analysis and Training Strategies in Face Recognition



■ **Figure 1** Average face verification accuracy across five popular datasets (LFW [9], AgeDB [17], CFP-FP [22], CALFW [36] and CPLFW [35]) for cross-resolution and equal-resolution (left). One example image pair for both scenarios in five image resolutions (right).

Celeb Dataset (MS1M) [6] with up to 87k identities. These networks are trained to map a facial image, typically after head pose normalization, into a feature space, in which intra-class features distances are minimized, and inter-class feature distances are maximized.

In Figure 1, we show that image accuracy drops for lower image resolution. Hence, we argue that the learned features depend on the training image resolution. Popular approaches learn a projection into a distinct feature space with datasets containing mainly high resolution (HR) images. However, in real-world applications, the image quality is often inferior. Besides external factors like illumination or the subject’s distance to the camera, sensor characteristics or image compression affect the image quality. For example, surveillance cameras capture faces at very low resolutions, in contrast to very high-quality mug-shots-like passport images. Another example is social media, which tries to recognize HR faces in the foreground and tiny low resolution (LR) faces in the background. In this work we focus on the most important characteristic of image quality - the image resolution.

LR face recognition [13, 2, 1] addresses the verification and identification of faces on images with the same coarse resolution. However, in real-world scenarios, the image resolution is arbitrary and unknown. Cross-Resolution (CR) face recognition addresses this problem of comparing images with varying resolutions, but has yet found minor attraction by the research community.

In this work, we first investigate the verification performance of a state-of-the-art face recognition network [3] on different image resolutions. We differentiate between CR and LR verification scenarios in our analysis. Figure 1 demonstrates that the performance is significantly worse in CR and LR scenarios across several datasets. At resolutions below  $10 \times 10$  px, the accuracy is slightly above 50%, which is only barely above guessing. Therefore, we assume a possibility for improvements, especially for very low image resolutions.

A major drawback of the works [5, 18] in CR face recognition is their focus on one specific image resolution, which assumes the image resolution to be known. Moreover, one needs several models to face a wide range of image resolutions, which are likely to occur in real-world applications. Zeng et al. [32] use a mix of two/four different image resolutions during training.

Our work distinguishes between two-resolution (i.e., training a network specifically with images in high resolution and one particular low resolution) and multi-resolution (i.e., feeding the model with HR and multiple LR images) training.

In summary, our main contributions are:

- We analyze the susceptibility for different image resolution on face verification in-depth.
- We propose two intuitive, straightforward approaches and show performance improvements on several datasets for CR face verification, especially at very low image resolutions.
- Lastly, we propose and publish three evaluation protocols to measure face recognition robustness against CR images. That is, to the best of our knowledge, the first benchmark for CR.

## 2 Related Work

### 2.1 Generic Face Recognition

In recent years, face recognition research has focused on loss functions applied mainly on ResNet [7] architectures. In [14], the authors propose an angular softmax loss with a multiplicative angular margin and in [27] an additive cosine margin. Deng et al. [3] applied an additive angular margin loss function, which can effectively extend the discriminating power of features. Recently, Kim et al. [12] presented with GroupFace a novel architecture that utilizes multiple group-aware representations to improve the quality of the features. Wang et al. [28] proposed a hierarchical pyramid diverse attention network. Schroff et al. [21] introduced the triplet loss to maximize the distances between an anchor image and its genuine sample (same identity) while minimizing the distance between an anchor image and its imposter sample (different identity).

### 2.2 Image Resolutions

To the best of our knowledge, no large training dataset provides different resolution versions of the same facial image. Furthermore, large datasets are often crawled from the web, and thus they lack very LR images on which the identity is unrecognizable. However, such a dataset is crucial to train a network, which is robust against varying image resolutions. The generation of LR images from HR images is an essential component in super-resolution. According to Zhou and Süsstrunk [37], a mapping from LR to HR images is often learned by synthetically downsampled HR images to retrieve target-oriented training data. They further state that the frequently used bicubic interpolation [10] significantly differs from real-world camera-blur and is not optimal. Nevertheless, simple bicubic downsampling is a cheap, reproducible, and effective way to lower the image resolution.

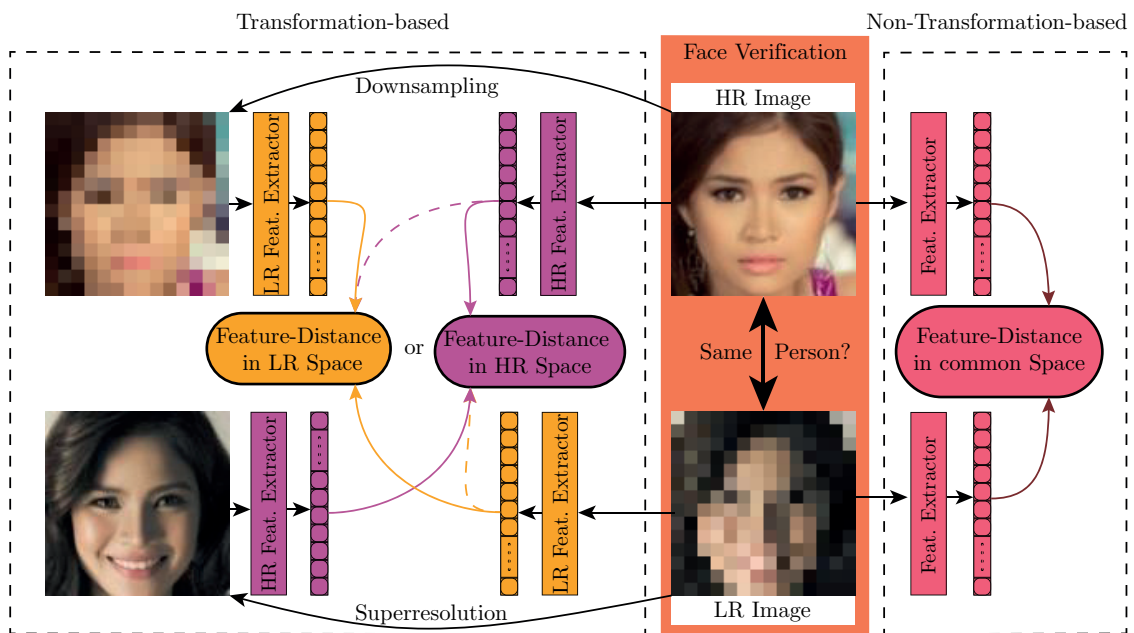
### 2.3 Cross-Resolution Face Recognition

According to [23], existing CR approaches can be grouped into two areas: 1) Transformation-based methods [34, 4, 19, 8] aim to transform images/features from low resolution to high resolution or vice versa to project them in a common space. 2) Non-transformation-based approaches [16, 32] intend to extract scale-invariant features directly into a common feature space. Wang et al. [29] show an exhaustive review of those methods for addressing CR face recognition, and Figure 2 gives a brief functional overview of those two methods.

#### Transformation-based Methods

Lu et al. [15] presented a deep coupled ResNet model containing one trunk network and a two-branch network. The trunk network extracts features, whereas the two-branch network transforms HR and the corresponding LR features into a space in which their difference can be minimized.

Zangeneh et al. [31] proposed a two-branch deep convolutional neural network. While the LR branch consists of a super-resolution network combined with a feature extraction network, the HR branch is only a feature extraction network. Both branches are trained in three different



■ **Figure 2** Transformation-based approaches (left) transform either learned image features (dashed path) or images into a shared space (solid path). Non-Transformation-based (right) methods aim to project scale-invariant image features directly.

training phases. In the benchmark, images are assigned to a particular branch depending on their resolution. A similar approach was used in [11]. They trained a U-Net with a combination of reconstruction and identity preserving loss to super-resolve multi-scale LR images. For feature extraction, they utilized a pretrained Inception-ResNet.

The authors of [25] proposed a coupled GAN network structure, which comprises two sub-nets, one for high resolution and one for low resolution. The correlation between the sub-net-generated features is maximized. Moreover, they considered facial attributes by implicitly matching facial details for both resolutions.

## Non-Transformation-based Methods

In [32], Zeng et al. presented a resolution-invariant deep network and trained it directly with unified LR and HR images. However, they used only resolutions in the range of 24 to 60 pixels for LR images.

Massoli et al. [16] proposed a student-teacher network approach. They showed that their approach can be more effective rather than preprocessing images with super-resolution techniques.

The authors of [18] report that their deep CNN architecture can address the problem of CR face recognition. They present a two-branch network architecture, which is trained in a pair-wise manner with multiple classification and contrastive loss functions.

In [5], Ge et al. focused on low computational costs in LR face recognition. Therefore, they introduced a new learning approach via selective knowledge distillation. A two-stream technique (large teacher model and a light-weight student model) is employed to transfer selected knowledge from the teacher model to the student model.



## 3 Experimental Setup

### 3.1 Baseline Network

As our baseline network, we choose a network structure comprising a modified ResNet-50 [7] as proposed in ArcFace [3], pretrained on ImageNet [20], and an ArcFace layer for classification.

The backbone network (ResNet-50) consists of a set of stacked residual blocks, which are repeated four times and contains in total 50 convolutional layers. It squeezes the input image from  $112 \times 112 \times 3$  px down to  $4 \times 4 \times 2048$  px utilizing multiple convolutions. After flattening the output from the backbone network, dropout is added. A bottleneck layer (512-dimensional fully connected layer), which represents the extracted features is added following [30, 14, 27]. Finally, a fully connected layer with 87k (number of identities in our training set) neurons is added. We then apply Additive Angular Margin Loss to the network following [3].

For training, we select the cleaned Microsoft MS1M [6] dataset containing about 5.8M images from about 87k identities. We perform random brightness and saturation variations, left-right flipping, and random cropping of images as data augmentation. All training parameters are set according to [3] except for a smaller batch-size of 128 due to hardware limitations. The learning rate is set to 0.01 and is decreased by a factor of 10 after epoch 9 and epoch 13. In total, we train for 16 epochs with momentum SGD optimizer. The dropout rate and weight decay are set to 0.5 and  $5 \cdot 10^{-4}$ , respectively.

### 3.2 Testing Datasets

We select five popular dataset (cf. Table 1) for evaluating face verification performance.

■ **Table 1** Statistics for five popular test datasets.

	LFW [9]	AgeDB [17]	CFP-FP [22]	CALFW [36]	CPLFW [35]
Identities	5749	568	500	5749	5749
Images	13233	16488	7000	13233	13233
Pairs	6000	6000	7000	6000	6000

We use the aligned face, which is cropped to  $112 \times 112 \times 3$  px afterwards for all testing datasets mentioned in Table 1. In this paper, we exclusively deal with images having equal width and height. For the sake of simplicity, we denote the image resolution by naming only the first dimension, i.e., a resolution of 112 px defines a  $112 \times 112 \times 3$  px image.

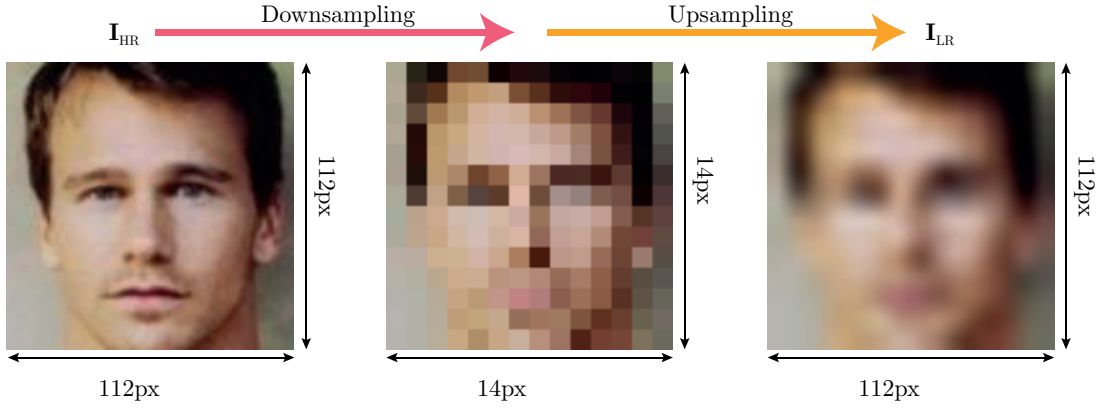
### 3.3 Reduction of Image Resolution

The baseline network requires HR input images  $\mathbf{I}_{\text{HR}}$  of the dimension 112 px. We simulate a resolution reduction by performing the following two steps: 1) Downsample  $F_{\text{down},r}(\cdot)$  images to an image dimension  $r$  in pixels followed by 2) upsampling  $F_{\text{up},r}(\cdot)$  the images back to the original image dimension and denote the resulting LR images as  $\mathbf{I}_{\text{LR}}$ . The complete process can be formulated as follows:

$$\mathbf{I}_{\text{LR}} = F_{\text{up},112}(F_{\text{down},r}(\mathbf{I}_{\text{HR}})) \quad (1)$$

For both sampling processes, bicubic interpolation [10] is applied. To reduce unwanted artifacts, typically introduced by the downsampling process, standard anti-aliasing techniques are employed. In subsection 4.1, we further investigate these effects.

## 01:6 Image Resolution Analysis and Training Strategies in Face Recognition



■ **Figure 3** Bicubic down- and up-sampling process to reduce the image resolution but keeping the image dimension.

Figure 3 illustrates the synthetic image resolution reduction. The left image  $\mathbf{I}_{HR}$  is a sample taken from the MS1M dataset with a resolution  $r = 112$ . In the center, the downsampled image  $F_{\text{down},14}(\mathbf{I}_{HR})$  with image dimension  $r = 14$  is depicted. Finally, the upsampled image  $\mathbf{I}_{LR}$  is shown on the right and has qualitatively considered an image resolution of  $r = 14$  but technically the same image dimension as the  $\mathbf{I}_{HR}$  image. It is evident that all the high-frequency information is removed by this synthetically resolution reduction. Simultaneously, the image dimension is equal to the original image, which is the required image dimension for our networks.

### 3.4 Accuracy in Face Verification

We report accuracy in all experiments, which denotes the face recognition rate in terms of face verification. To calculate the accuracy value for a given dataset, we first take the cosine distances  $d$  between features of every image pair  $(\mathbf{I}_1, \mathbf{I}_2)$  extracted from a model  $M(\cdot)$  according to  $N$  image pairs defined in the specific evaluation protocol for each dataset. respectively:

$$d = 1 - \frac{M(\mathbf{I}_1) \cdot M(\mathbf{I}_2)}{\|M(\mathbf{I}_1)\|^2 \|M(\mathbf{I}_2)\|^2} \quad (2)$$

Then, we use 10-fold cross-validation to find optimal thresholds that can separate feature distances of genuine from imposter pairs. The number of correctly identified genuine and imposter samples from a total number of samples  $N$  are then named as true positives  $TP$  and true negatives  $TN$ , respectively. We then calculate an accuracy score  $Acc$  as follows:

$$Acc = \frac{TP + TN}{N} \quad (3)$$

For all experiments in the CR scenario we generate two evaluation protocols by flipping the pairwise matching resolution from

$$\left( M(F_{\text{up},112}(F_{\text{down},r}(\mathbf{I}_1))), M(\mathbf{I}_2) \right)$$

to

$$\left( M(\mathbf{I}_1), M(F_{\text{up},112}(F_{\text{down},r}(\mathbf{I}_2))) \right)$$

We then calculate the accuracy score for both test datasets and then compute the mean.



## 4 Analysis of Image Resolution Susceptibility

In this section, we first investigate the effect by reducing the resolution across five test datasets. Then, we examine the performance of the baseline network under LR conditions in CR and ER scenarios. Afterward, we take a closer look at the extracted features, especially at the cosine distance between the image pairs, which is used to classify them as genuine or imposter.

### 4.1 Resolution Reduction on several Datasets

To better understand what happens when performing resolution reduction synthetically, we analyze the effect of downsampling on several testing datasets. Hence, we calculate a mean image across the whole dataset and highlight the mean pixel difference between LR and HR images. The mean images in Figure 4 are computed as follows:

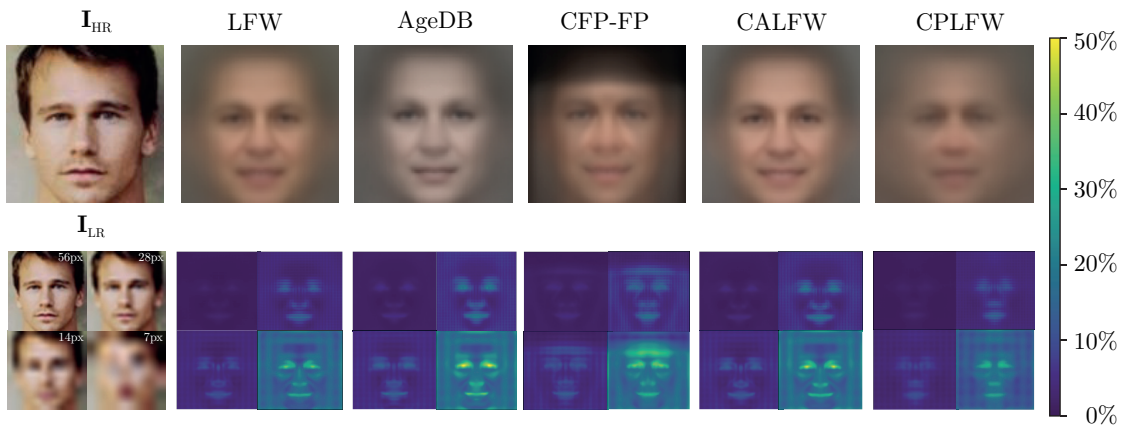
$$\mathbf{I}_{\text{HR}}^{\text{mean}} = \frac{1}{N} \sum_{i=1}^N \mathbf{I}_{\text{HR},i} \quad (4)$$

where  $N$  denotes the number of elements of the dataset.

Below each mean HR image, we denote the mean absolute pixel differences  $D_r$  between synthetically reduced images  $I_{LR,r}$ , and original  $I_{HR}$  images across each dataset. We retrieve those images for four resolutions  $r \in \{7, 14, 28, 56\}$  according to:

$$\mathbf{D}_r^{\text{mean}} = \frac{1}{N} \sum_{i=1}^N \left( \left| \mathbf{F}_{\text{up},112}(\mathbf{F}_{\text{down},r}(\mathbf{I}_{\text{HR},i})) - \mathbf{I}_{\text{HR},i} \right| \right) \quad (5)$$

As expected, the resolution reduction process in all datasets is heavily affected by eye, nose, and mouth regions. High detail information in those regions is lost. This reasonably results in worse face verification performance as we show later in the next section. The maximum derivation of a single LR image pixel concerning its corresponding pixel in the HR image is about 50%. In all pixel-difference images grid-style artifacts occur, which in our opinion result from the anti-aliasing method of the bicubic interpolation algorithm.



**Figure 4** The left column shows a high resolution sample image  $\mathbf{I}_{\text{HR}}$  from MS1M and its corresponding reduced-resolution images  $\mathbf{F}_{\text{down},r}(\mathbf{F}_{\text{up},112}(\mathbf{I}_{\text{HR}}))$  for four resolutions  $r \in \{7, 14, 28, 56\}$ . In the first row are then the mean images  $\mathbf{I}_{\text{HR}}^{\text{mean}}$  for several datasets shown. Below are the pixel difference images  $\mathbf{D}_r^{\text{mean}}$  for specific resolutions.

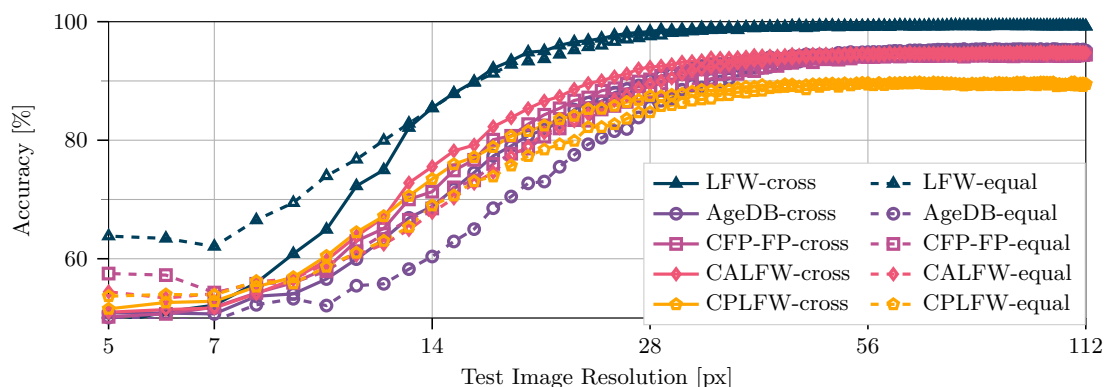
The mean dataset HR images are quite different across all datasets. Pose variations in CPLFW dataset result in blurrier areas of the image. In contrast, the CALFW and LFW dataset images seem to be very accurately aligned and show almost a clear and detailed average face. Interestingly, the background in the CFP-FP dataset is very dark compared to other datasets. This results from cropped faces which are padded in black, especially in the top image region. Also, the pose variation is visible in the average face. Some ghosting effects are also present in that image.

The mean absolute pixel difference images show the same pattern across all datasets. With decreasing resolution the difference is more visible, especially in the high frequency regions (eyes, nose, and mouth).

## 4.2 Face Verification Accuracy

As depicted in Figure 5, the performance on all datasets drops for lower resolutions as expected. The accuracy on the LFW dataset is best for high resolution but drops heavily for lower resolutions. The worst performance on high resolutions can be seen on the CPLFW dataset. A reason for this behavior can be the large pose variations in this test set, which are not occurring in the training set and therefore unknown to the network.

Interestingly, we see different decreasing characteristics between the CR and ER scenarios. To a particular resolution, all datasets show worse performance in the ER scenario than in the CR scenario. This performance gap is reasonable since more pixel information is present in a CR pair than in an ER image pair. Against intuition, this trend reverses for very low resolutions except for the AgeDB dataset. We explain this behavior with a more significant domain shift for the network necessary within the CR image pairs than within the ER image pairs. Our network is familiar with HR images, and down to a specific resolution, it can interpret lower quality faces quite well. Whereas beneath a threshold, both LR images are unfamiliar to the network, and thus, features represent different ID characteristics compared to HR features. However, in the AgeDB dataset is a significant age gap within the pairs, which implicates that on LR images, for example, large hair-style variations or the effect of gray-scale vs. color images, might confuse the network for positive image pairs.



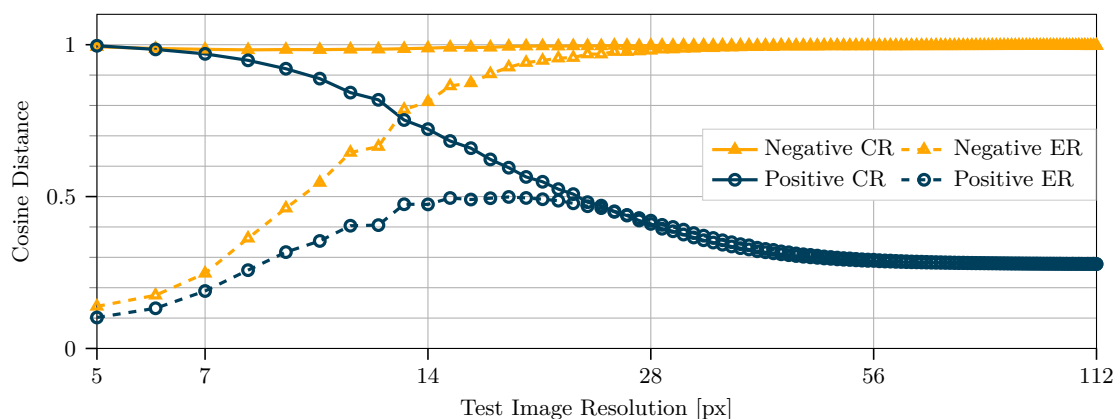
■ **Figure 5** Face verification accuracy across several datasets for different image resolutions in cross-resolution (high resolution vs. low resolution image) and equal-resolution (low resolution vs. low resolution image) scenario.

### 4.3 Feature Distances

Since the accuracy depends on a distance threshold, which classifies sample pairs as genuine or imposter, the distance between both features vectors is crucial for the verification accuracy. Hence, we look at the average feature distance for all genuine and imposter image pairs of the LFW dataset (cf. Figure 6). We divide the diagram roughly into three sections: 1)  $r > 60$  px, 2)  $> 20$  px  $< r < 60$  px, and 3)  $r < 20$  px. In the first section, feature distances between genuine and imposter image pairs seem to be independent of the image resolution. The average distance is about 0.3 within genuine pairs and 1.0 within imposter pairs, which means that the high dimensional feature vectors are almost orthogonal. The second section reveals, that in both CR and ER scenario the distance of genuine image pairs tends to increase, whereas the distance for imposter image pairs is only slightly decreasing. A small reduction of image resolution causes repelling features. However, reducing the image resolution more (section 3), all LR image features are projected closely together (far away from HR features), which results in small distances for ER and high distances in the CR scenario. Considering that almost all pairs are then categorized as genuine (in the CR scenario) or imposter (in the ER scenario), the face verification performance is merely guessing.

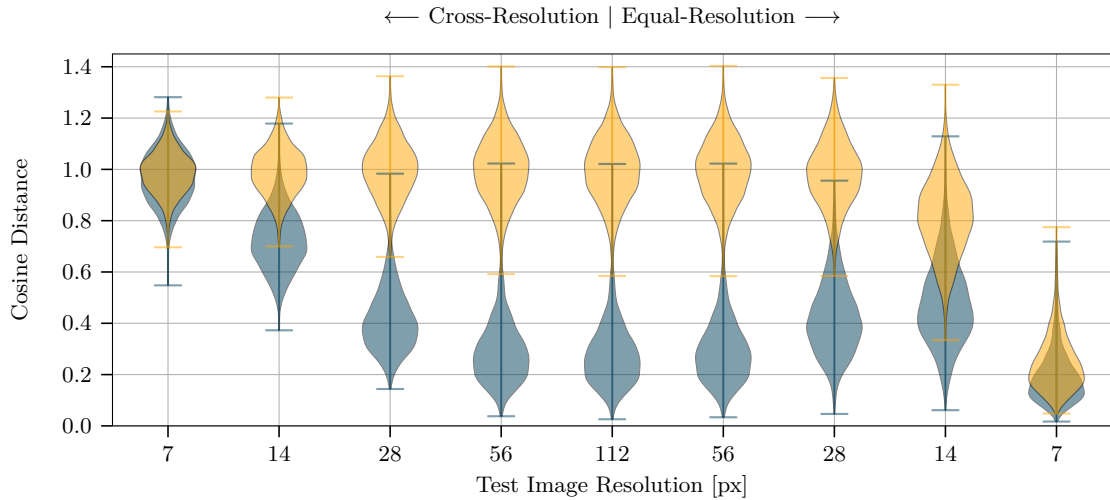
For the CR scenario, we conduct that our network is not able to extract accurate features for the very LR images. Hence, this results in a large distance between features because the HR image features are still very distinctive. However, in the ER scenario, both images are unfamiliar to the network, which results in resembling extracted features.

Figure 7 captures the cosine feature distance distributions for the LFW dataset. The center violin plots represent the feature distance distribution for HR image pairs. Distances for genuine and imposter image pairs are clearly distinguishable. The genuine distances are mainly in a range between 0.1 and 0.6, whereas imposter distances are mostly in the field of 0.6 and 1.4. Both classes can be separated effectively with a threshold of about 0.6, and thus, the accuracy for HR face verification is best (cf. Figure 5). On the left side, distributions for the CR scenario are shown, whereas on the right side, ER feature distributions are plotted. For images resolutions of 56 px and 28 px the distributions in both scenarios is similar to the HR distribution. The fact that the peak feature distance for genuine image pairs even exceeds the maximum distance for imposter pairs in the CR scenario at very low resolution 5 px, leads to the conclusion that image resolution



■ **Figure 6** Average cosine feature distances between image pairs for genuine (o) and imposter (Δ) pairs in the LFW dataset. Dashed lines shows distances in the equal-resolution scenario, while solid lines represent distances in the CR scenario.

## 01:10 Image Resolution Analysis and Training Strategies in Face Recognition



■ **Figure 7** Cosine feature distance distributions for genuine (blue) and imposter (yellow) cross-resolution (left) and equal-resolution (right) pairs in the LFW dataset. Five different resolutions are shown for our baseline model.

has a more significant impact on the distance than the identity itself. The gap between CR and ER accuracy for very low resolutions is therefore reasonable. Although the small distances for both kinds of image pairs in the ER case, more genuine feature distance still have a smaller value. This behavior explains a higher accuracy for very low resolutions in the ER scenario compared to CR scenario. Further experiments with CFP-FP, AgeDB, CALFW, and CPLFW datasets underline this trend.

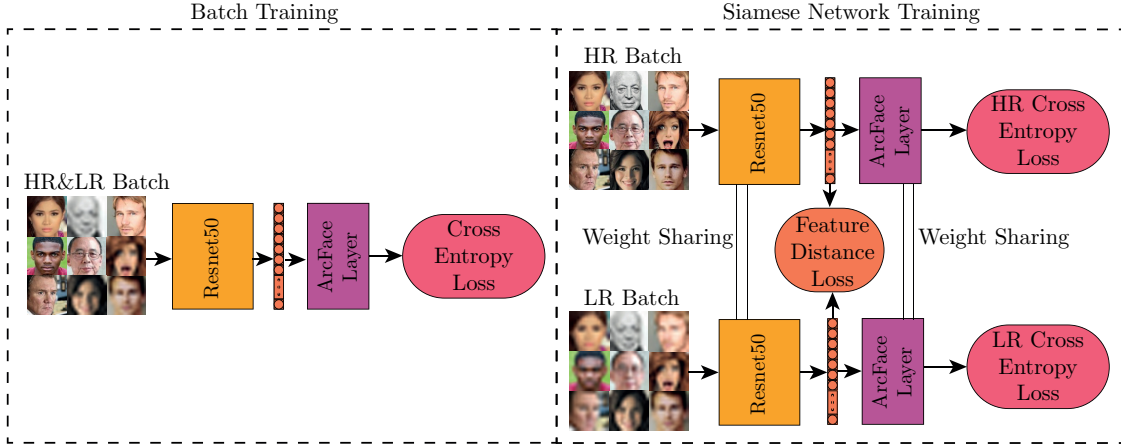
## 5 Training Methods

To improve the separability between feature distances of genuine and imposter image pairs, and hence, the accuracy, we pursue two intuitive non-transformation-based methods (cf. Figure 8): 1) CR batch training and 2) CR siamese network training.

In all training sessions, we use the MS1M dataset and train in total for 16 epochs. All training parameters are set according to our baseline (cf. section 3) for a fair comparison.

### 5.1 Cross-Resolution Batch Training

Motivated by [32, 16], we first propose a straightforward batch CR training approach to tackle the susceptibility to image resolutions. Instead of using HR images only, we randomly select half of the images per batch and synthetically reduce their resolution (cf. subsection 3.3). We train several specific networks specializing each on a particular resolution. For the sake of simplicity, we refer to these models according to the following rule:  $BT-r$  where  $r$  denotes the specific LR value during training. We apply resolutions  $r \in \{x \in \mathbb{N} \mid 5 \leq x \leq 22\} \cup \{28, 56\}$  in our experiments. Since we only take half of the images per batch for resolution reduction, the network is still exposed to HR images and thus learns to extract features from HR and LR images at the same gradient update step.



■ **Figure 8** Overview of our proposed methods. The left part shows the cross-resolution batch training method, whereas the right part shows the siamese network cross-resolution training approach.

## 5.2 Siamese Network Cross-Resolution Training

Inspired by Tang et al. [26], we implement a siamese network structure (cf. Figure 8 CR training). Each branch of the network consists of our baseline architecture and trains the network for a specific image resolution. With weight-sharing across all branches, we keep the same number of parameters and ensure similar inference during test-time compared to  $BT-r$ . We construct two branches for training with exactly two resolutions (the high resolution and one low resolution). Our objective is to closely project the corresponding features from all branches for a specific image with an arbitrary resolution. We add a new loss function to the network to enforce this, which penalizes a high cosine distance between both features. We employed the cosine distance metric to match the evaluation protocol. Applying a HR image to the ArcFace network (HR branch),  $f_{HR}$  then denotes the corresponding output feature vector, and images from the particular LR branches are named  $f_{LR}$  accordingly. The cosine feature distance loss  $\mathcal{L}_{dist}$  is then calculated as:

$$\mathcal{L}_{dist} = 1 - \frac{f_{HR} \cdot f_{LR}}{\|f_{HR}\|^2 \|f_{LR}\|^2} \quad (6)$$

For both branches, we calculate the cross-entropy loss  $\mathcal{L}_{CE,HR}$  and  $\mathcal{L}_{CE,LR}$ , respectively. We weigh all three losses approximately equally and multiply the feature distance loss by a factor of 25. Finally, we conclude the total loss function  $\mathcal{L}$  for the siamese training approach as follows:

$$\mathcal{L} = \mathcal{L}_{CE,HR} + \mathcal{L}_{CE,LR} + 25 \cdot \mathcal{L}_{dist} \quad (7)$$

Due to the siamese network architecture, both images, in high resolution and low resolution, need to be propagated through each branch. Thus, the training time is about double in the two resolution training scenario. In our experiments, we select the following resolutions  $r \in \{5, 6, 7, 8, 12, 14, 20, 28, 56\}$  to train specific resolution models. In the following, we refer to this training technique as  $ST-r$ .

## 6 Experimental Results

In this section, we present and discuss the results of our proposed approaches. Firstly, we focus on the two-resolution scenario, i.e., high resolution (112 px) and one specific low resolution. Secondly, focus on simultaneously training with multiple image resolutions, i.e., high resolution (112 px) and

multiple low resolutions (7 px, 14 px, 28 px, and 56 px) in one training. We analyze the accuracy on five popular datasets and compare the distances of the resulting features for all methods. Moreover, we introduce a new evaluation protocol to measure the performance of a model for multiple resolutions in the test dataset. We conclude this section with a comparison of all methods proposed in this paper, especially concerning the differences in accuracy and training time.

## 6.1 Two-Resolution Training Scenario

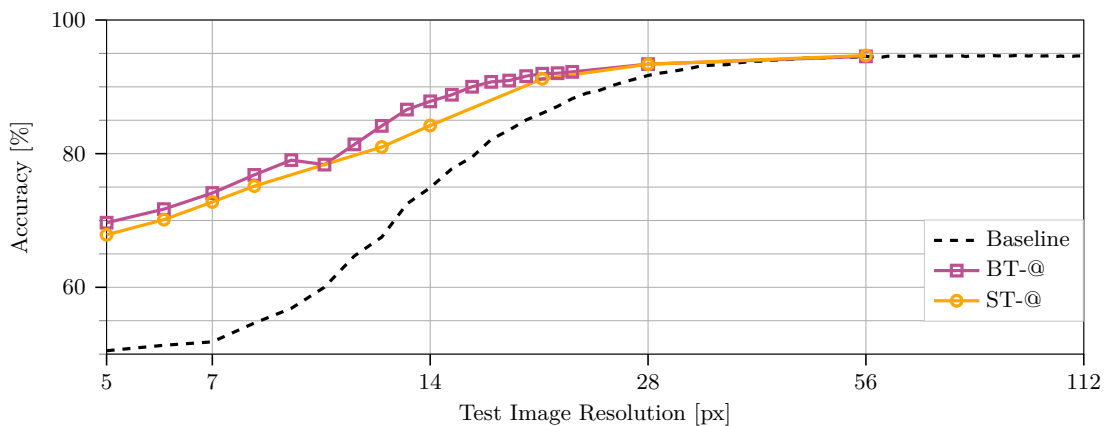
As previously mentioned in section 5, we now analyze the CR batch training approach  $BT-r$  and the siamese network CR training approach  $ST-r$  concerning the face verification accuracy on five popular datasets. This two-resolution training scenario trains each model with exactly two specified resolutions and compares the results to the baseline network concerning accuracy and feature distances.

### Face Verification Accuracy

As introduced in subsection 3.4, accuracy is a standard metric to measure the performance of a face verification model. Figure 9 depicts the average face verification accuracy across five common datasets of the  $BT-r$  and  $ST-r$  model compared to our baseline model. Note that  $BT-@$  and  $ST-@$  data points represent different models trained explicitly for the test resolution. Both approaches outperform the baseline model for low image resolutions. For very low resolutions, i.e., 5 px to 8 px, the performance can be increased from  $\approx 50\%$  up to 70%. Above  $r \approx 40$  px, no significant difference exists between all approaches, which affirms our expectations since the LR images are visually hardly distinguishable from the original images, and the absolute pixel difference is minimal (cf. subsection 3.3).

Generally, the performance improvement is increasing with decreasing resolutions. The  $BT-r$  method performs slightly better than the  $ST-r$  method, from which we conclude that the siamese approach might concentrate too much on projecting the features of the same image in different resolutions into the same space than on classifying the correct identity regardless of the resolution. For applications with a known fixed resolution, a  $BT-@$  are the better choices.

Moreover, we compare our results on the very popular LFW dataset with two other approaches (cf. Table 2): First, the selected knowledge distillation technique proposed by Ge et al. [5], and second, the attribute-guided coupled GAN approach introduced by Talreja et al. [25]. Our systems



■ **Figure 9** Evaluation of average face verification accuracy across five popular datasets for different resolution with several models.

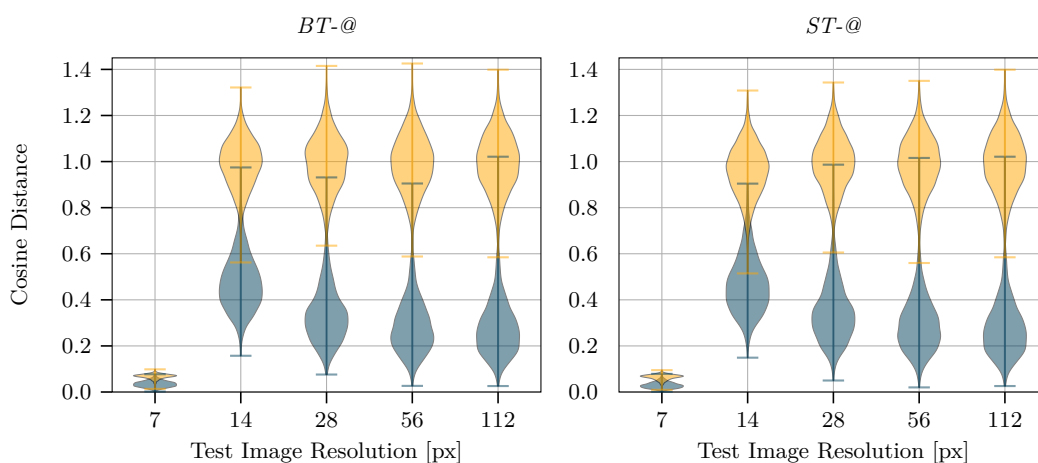
■ **Table 2** Face verification accuracy on the LFW dataset. The best performance of each image resolution is marked bold.

Image Resolution	Model	Accuracy
64 px	<i>BT-64</i> (ours)	<b>99.38%</b>
	<i>ST-64</i> (ours)	99.35%
	S-64-sc [5]	92.83%
	Talreja et al. [25]	94.92%
32 px	<i>BT-32</i> (ours)	<b>99.08%</b>
	<i>ST-32</i> (ours)	98.32%
	S-32-sc [5]	89.72%
	Talreja et al. [25]	91.08%
16 px	<i>BT-16</i> (ours)	<b>98.17%</b>
	<i>ST-16</i> (ours)	97.8%
	S-16-sc [5]	85.87%

clearly outperform both competitors. However, the comparison to Ge et al.’s approach is not fair. Their baseline model (teacher model) only reaches an accuracy of 97.15%, which is not comparable to our baseline and state-of-the-art. On the other hand, the model’s number of parameters also differs. They only trained both models for three different resolutions, showing only a few snapshots and not the whole performance curve. The lowest resolution, (16 px) is relatively high compared to our analysis, so we cannot fully exploit our strengths here.

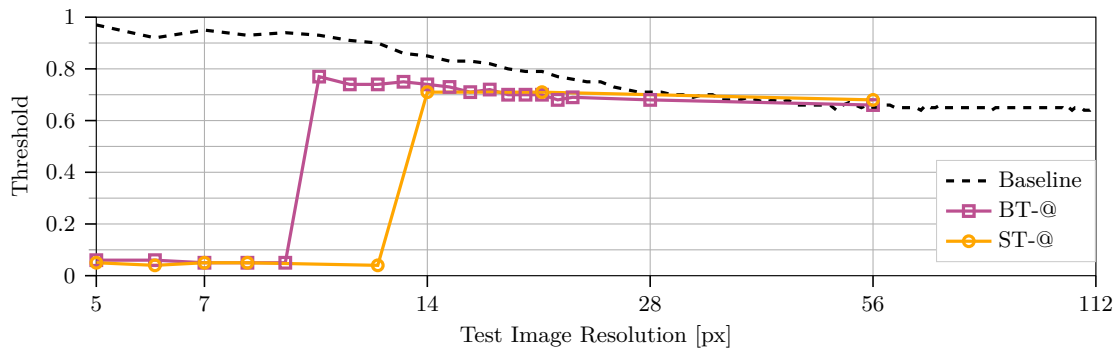
## Feature Distances

Similar to subsection 4.3, we pick five different resolutions and take a closer look at the features themselves. To be more precise, we plot the distance distributions for genuine and imposter image pairs from the LFW dataset.



■ **Figure 10** Cosine feature distance distributions for genuine (blue) and imposter (yellow) cross-resolution pairs in the LFW dataset. For both models, @ denotes that the training resolution matches the test resolution.





■ **Figure 11** Best thresholds selected for calculating the accuracy on the LFW dataset using different models. In the model description an @ denotes, that the training resolution matches the test resolution.

Figure 10 shows that distances of genuine and imposter pairs are much better separable for low resolutions 14 px, 28 px and 56 px than the baseline results (cf. Figure 7). The main difference compared to the baseline is the shift of genuine and imposter distances to a range of almost 0 and 0.1 in the very low resolution scenario (7 px). This behavior is remarkable and shows that both networks learn to project features from very different resolutions into the same space. Although all distances are small, imposter distances are still greater than genuine, and the distributions are separable, consistent with the accuracy improvement discussed in the previous subsection (cf. Figure 9). Furthermore, there is no significant difference between both proposed methods. This is in line with with the last section’s accuracy values.

To understand and determine the exact resolution where the feature distances drop so much, we calculate the optimal threshold and analyze the corresponding accuracy values (cf. subsection 3.4). Figure 11 depicts the threshold values for the baseline,  $BT-r$ , and  $ST-r$  models on all tested image resolution in the CR scenario. Thresholds for our baseline model are increasing for lower resolutions. This trend is consistent with our results in subsection 4.3, where genuine and imposter feature distances increase for lower resolutions. Our two-resolution training networks show a significant drop at  $r = 9$  px for  $BT-r$  and  $r \approx 12$  px for  $ST-r$ . From these points on, both models behave differently in the training sessions and project features for significant resolution differences more closely.

## 6.2 Multi-Resolution Training Scenario

We propose multiple-resolution training for both approaches to simulate a more applicable model, which is capable of handling arbitrary resolutions. We train the  $BT-r$  model with more than two resolutions simultaneously by randomly picking a different resolution in  $\{7, 14, 28, 56, 112\}$  to generate a LR image. Each batch contains HR images and multiple LR images with different resolutions. We find that those five resolutions equally represent the range of image resolutions. This range reflects, for example, equivalent distances from subjects to the camera in real life. The probability of each resolution is set to be equal. We name this approach  $BT-M$  in the following.

In the  $ST-r$  approach, we apply two different methods for training with multiple resolutions simultaneously. First, for the LR branch, we randomly pick a resolution from the numbers  $\{7, 14, 28, 56\}$  and feed the LR branch with LR images of different resolutions within each batch. The HR branch always takes 112 px images. This training with in total five different resolutions simultaneously doubles the training time. In the following, this approach will be referred to as



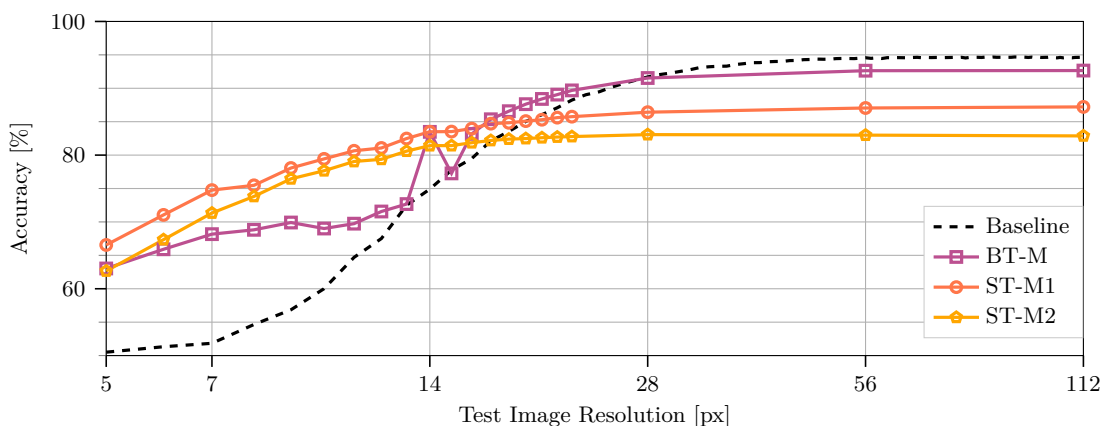
*ST-M1*. The second method *ST-M2* extends the siamese network to five branches, each branch representing a particular defined resolution (7 px, 14 px, 28 px, 56 px, and 112 px). Consequently, four feature distance losses are calculated each between the HR and the corresponding LR branch. Moreover, we also calculate the cross-entropy loss for each branch. All feature distance losses are weighted each with a factor of 25, to be in the same order of magnitude as the cross-entropy losses. The training time for this experiment is about five times longer than the baseline because it is scaling with the number of defined resolutions for training.

## Face Verification Accuracy

Figure 12 presents the face verification accuracy for *BT-M*, *ST-M1*, and *ST-M2* model across arbitrary image resolutions. All three approaches perform significantly better than the baseline model in resolutions below 13 px and worse above a resolution of 28 px. Note that there is a significant peak at a resolution of 14 px, especially for *BT-M*. One reason for this could be that this specific resolution was used during training, and hence, this effect is also visible at resolutions 7 px, 28 px, and 56 px.

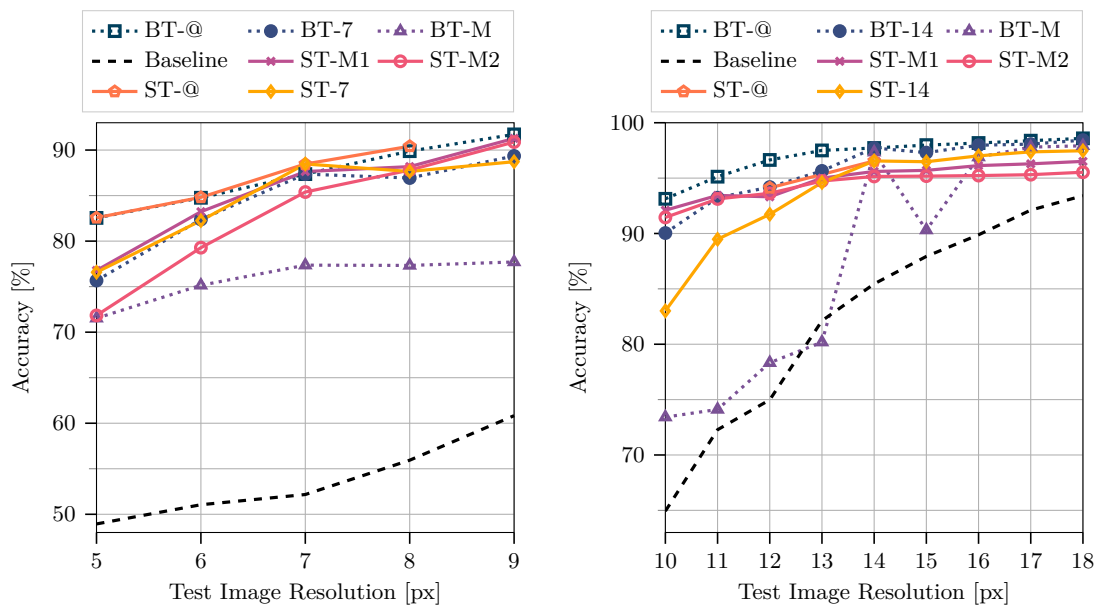
Another finding is that the siamese network CR training outperforms the CR batch training for low resolutions ( $r < 16$  px) and vice-versa for mid and high resolutions ( $r > 16$  px). For  $r = 7$  px, the *ST-M1* model achieves an accuracy score of  $\approx 75\%$ , which is almost 25% above the baseline performance and even higher than *ST-7*. At the same time, that approach loses about 8% performance at high resolutions  $r = 56$  px. For a more scale-comprehensive performance score, we will introduce three new evaluation protocols in subsection 6.3.

Figure 13 investigates the performance at and close to two selected resolutions, 7 px and 14 px. On the left side, we can see that *BT-7* and *ST-7* optimized the performance strictly for the 7 px resolution, and hence they perform worse in the neighboring regions. *BT-@* and *ST-@*, which represent specific resolution trained models, perform best at each scale, and this is reasonable due to the training with that particular image resolution. The performance loss for all multiple-resolution trained approaches (*BT-M*, *ST-M1*, and *ST-M2*) is compensated by the benefit of having a single model for arbitrary resolutions. The right part of Figure 13 shows an excerpt of resolutions from 10 px to 18 px. Here, the wave effect of *BT-14* and *ST-14* is also slightly visible, meaning that those two models perform relatively best on exactly 14 px resolution.



■ **Figure 12** Average face verification accuracy across five popular datasets for different image resolutions with several models. Except for the Baseline all models were trained using multiple image resolutions.

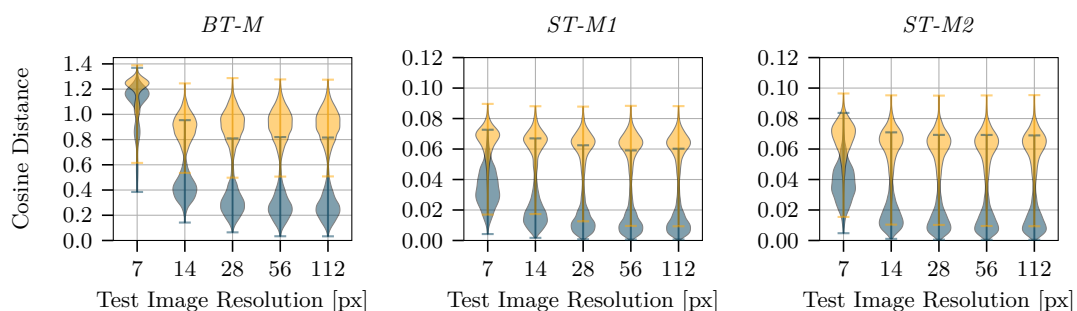
## 01:16 Image Resolution Analysis and Training Strategies in Face Recognition



■ **Figure 13** Accuracy on the LFW dataset for several models trained with different image resolutions. In the model description an @ denotes, that the training resolution matches the test resolution.

### Feature Distances

Interestingly, in terms of feature distance distributions (cf. left part of Figure 14), the multi-resolution batch training is not behaving similarly to the two resolution batch training. Specifically for  $BT-r$ , at very low resolutions ( $r = 7$  px), the feature distance distributions for genuine and imposter pairs are even larger than for all other resolutions. This characteristic fits to the  $BT-r$  accuracy at that scale (cf. Figure 13). In contrast to the two-resolution case, both siamese training approaches ( $ST-M1$  and  $ST-M2$ ) project features for all resolutions closer together. We conduct this from very low distances across all scales (center and right parts in Figure 14). The maximum feature distance for both approaches is about 0.1.



■ **Figure 14** Cosine feature distance distributions for genuine (blue) and imposter (yellow) cross-resolution pairs in the LFW dataset at different test-set image resolutions. For training, multiple image resolutions were used.

### 6.3 Evaluation Protocols for Multiple Resolutions

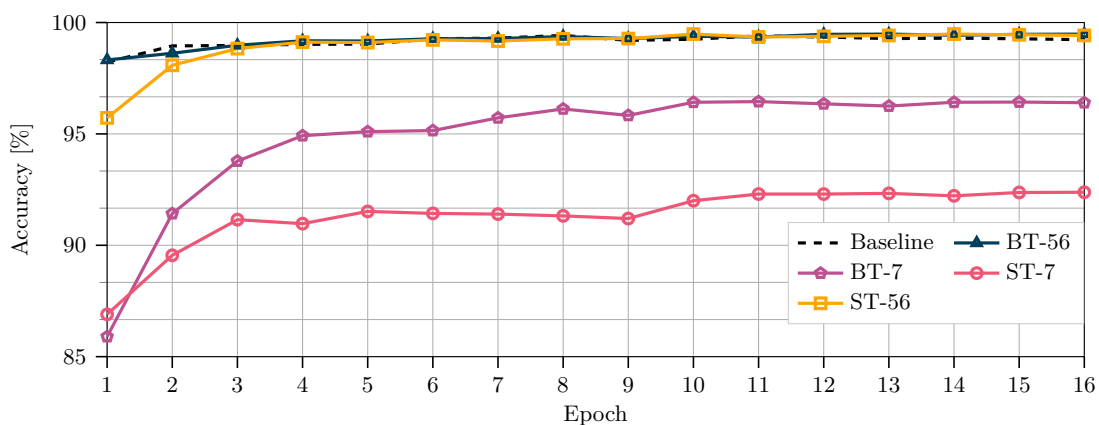
Evaluation protocols for common public datasets are not taking into account the image resolution. In the previous sections, we only considered a specific image resolution to calculate the face verification accuracy. With single networks (cf. subsection 6.2) capable of handling arbitrary image resolutions at once, there is a need for a more meaningful evaluation considering multiple resolutions. Therefore, we propose specific evaluation protocols for all five datasets, with a focus on four different resolution ranges:

- Low resolutions:  $r \in \{x \in \mathbb{N} \mid 5 \leq x \leq 10\}$
- Mid resolutions:  $r \in \{x \in \mathbb{N} \mid 11 \leq x \leq 40\}$
- High resolutions:  $r \in \{x \in \mathbb{N} \mid 41 \leq x \leq 112\}$
- All resolutions:  $r \in \{x \in \mathbb{N} \mid 5 \leq x \leq 112\}$

The evaluation protocols define the resolution for each image in each pair for the corresponding dataset, and we keep the probability for each resolution in the generation process equal. All protocols are available at: <https://github.com/martlgap/btm-stm>.

### 6.4 Comparison of the proposed Methods

To conclude this chapter, we provide a comparison between all introduced methods. First, we analyze the verification performance on HR images for all proposed methods and compare them to the baseline approach. Figure 15 shows the accuracy of the LFW dataset for each epoch. We select models *BT-7*, *BT-56*, *ST-7*, and *ST-56* to represent both, shallow and relatively high resolutions. After the first epoch, our baseline model achieves about 98% accuracy, followed by almost peak accuracy already after the second epoch. During epochs 3 and 16, no significant changes in accuracy are visible. The *BT-56* starts with equal accuracy after the first epoch and then takes another two epochs to reach almost peak accuracy. The *ST-56* gets only after epoch 4 approximately peak performance. This model needs significantly more samples than both previously mentioned models to achieve similar accuracy. One reason could be the additional feature distance loss, which forces the network to minimize feature distances and learn a reasonable classification.



■ **Figure 15** Evaluation of face verification accuracy on the original LFW dataset for high resolution images.

■ **Table 3** Comparison of training time per epoch and accuracy on LFW dataset for different test image resolution protocols. Bold numbers denote the best performance across all methods.

Model	Training Time per Epoch [h]	Accuracy [%] for Test Resolution					
		112 px	all_res	high_res	mid_res	low_res	5 px
Baseline	2	99.23	96.86	99.20	95.89	77.57	54.65
BT-M	2	<b>99.30</b>	<b>97.72</b>	<b>99.33</b>	<b>97.78</b>	87.17	71.53
ST-M1	4	97.40	96.76	97.35	96.98	<b>91.50</b>	<b>76.78</b>
ST-M2	20	95.62	95.07	95.62	95.51	88.72	71.84

The peak performance for both methods *BT-7* and *ST-7* are significantly lower compared to the other approaches. This decrease could evolve from too little information in the shallow LR images, which might probably be just too little resolution to be able to learn a proper feature extraction. Moreover, models converge slower and need at least 10 epochs to reach the overall maximum accuracy region.

Second, Table 3 compares all presented methods to their training time per epoch and performance in the multi-resolution scenarios and depicts accuracy values on the LFW dataset. We conduct that compared to the two resolution techniques, both *ST-M1* and *ST-M2* models clearly outperform the baseline and the *BT-M* for low resolutions. Focusing on higher resolutions, we conclude that *BT-M* is the best performing method. Even for the original image resolution of 11 px, the *BT-M* model performs better than the baseline. We think this is reasonable because using lower resolutions additionally during training can be seen as extra data augmentation and hence, can improve the performance. One also has to compromise that for an absolute performance improvement of about 14% in the *low\_res* protocol, the performance for *high\_res* drops about 2%. In the second siamese training approach, *ST-M2* is performing worse in all categories than *ST-M1*. Therefore, we conclude that the much greater effort for training is not worth it. It seems to be less important to force a network to learn close features for the same image in different resolutions, within each batch, than across several batches.

Lastly, the number of parameters, and hence the inference time, is equal for all models, thus making the comparison fair and reasonable.

## 7 Conclusions and Future Work

This work analyzes the impact of different image resolutions on face verification performance utilizing a state-of-the-art approach. The distances between extracted features are investigated in detail. Our findings are that facial features extracted from established face recognition networks are not scale-invariant, and hence the performance decreases substantially for lower image resolutions.

To obtain the best performance, the resolution of the testing images must be the same as in the corresponding training dataset for the network. To overcome this problem, we propose two intuitive methods to learn scale-invariant features directly: 1) Training our network with batches containing an equal amount of LR and HR images. Experiments across five conventional test datasets show improvements up to 24.80% for very low image resolutions of 5 px. 2) Training a siamese network structure, which additionally minimizes feature distances between LR and HR versions of the same image besides the cross-entropy loss. Evaluations across five conventional test datasets indicate an improvement in performance up to 31.77%.

Furthermore, we train our proposed models with several resolutions at once. Hence, a single model can be applied to arbitrary image scales, making it more applicable. We also report a considerable improvement of 17.96% with our best model *ST-M1* for CR verification performance,

especially for low resolutions. Compared to the simple batch training method, the siamese network CR training performs better for low resolutions and worse for mid and high resolutions. For applications with a known fixed resolution, the latter method is the better choice.

Lastly, we introduce and release three different evaluation protocols for five popular datasets, defining multiple resolutions for CR scenarios.

Our work showed that a loss on feature distances helps to mitigate the resolution susceptibility in face verification. Therefore, in the future, we want to employ a specifically designed triplet loss variant, which minimizes intra-class and maximizes inter-class feature distances. We also want to extend the downsample process by using arbitrary blur kernels described in [33] and applying them in our work.

---

## References

- Omid Abdollahi Aghdam, Behzad Bozorgtabar, Hazim Kemal Ekenel, and Jean-Philippe Thiran. Exploring factors for improving low resolution face recognition. In *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*, pages 2363–2370. IEEE, 2019.
- Zhiyi Cheng, Xiatian Zhu, and Shaogang Gong. Low-resolution face recognition. *CoRR*, abs/1811.08965, 2018. [arXiv:1811.08965](#).
- Jiankang Deng, Jia Guo, Niannan Xue, and Stefanos Zafeiriou. Arcface: Additive angular margin loss for deep face recognition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 4690–4699, 2019.
- Berk Dogan, Shuhang Gu, and Radu Timofte. Exemplar guided face image super-resolution without facial landmarks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops*, pages 0–0, 2019.
- Shiming Ge, Shengwei Zhao, Chenyu Li, and Jia Li. Low-resolution face recognition in the wild via selective knowledge distillation. *IEEE Transactions on Image Processing*, 28(4):2051–2062, 2018.
- Yandong Guo, Lei Zhang, Yuxiao Hu, Xiaodong He, and Jianfeng Gao. Ms-celeb-1m: A dataset and benchmark for large-scale face recognition. In *European conference on computer vision*, pages 87–102. Springer, 2016.
- Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016.
- Chih-Chung Hsu, Chia-Wen Lin, Weng-Tai Su, and Gene Cheung. Sigan: Siamese generative adversarial network for identity-preserving face hallucination. *IEEE Transactions on Image Processing*, 28(12):6225–6236, 2019.
- E. G. Huang, G. B. Learned-Miller. Labeled Faces in the Wild: Updates and New Reporting Procedures. Technical Report UM-CS-2014-003, University of Massachusetts, Amherst, May 2014.
- Robert Keys. Cubic convolution interpolation for digital image processing. *IEEE transactions on acoustics, speech, and signal processing*, 29(6):1153–1160, 1981.
- Vahid Reza Khazaie, Nicky Bayat, and Yalda Mohsenzadeh. Ipu-net: Multi scale identity-preserved u-net for low resolution face recognition. *arXiv preprint*, 2020. [arXiv:2010.12249](#).
- Yonghyun Kim, Wonpyo Park, Myung-Cheol Roh, and Jongju Shin. Groupface: Learning latent groups and constructing group-based representations for face recognition. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 5621–5630, 2020.
- Pei Li, Loreto Prieto, Domingo Mery, and Patrick J Flynn. On low-resolution face recognition in the wild: Comparisons and new techniques. *IEEE Transactions on Information Forensics and Security*, 14(8):2000–2012, 2019.
- Weiyang Liu, Yandong Wen, Zhiding Yu, Ming Li, Bhiksha Raj, and Le Song. Sphereface: Deep hypersphere embedding for face recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 212–220, 2017.
- Ze Lu, Xudong Jiang, and Alex Kot. Deep coupled resnet for low-resolution face recognition. *IEEE Signal Processing Letters*, 25(4):526–530, 2018.
- Fabio Valerio Massoli, Giuseppe Amato, and Fabrizio Falchi. Cross-resolution learning for face recognition. *Image and Vision Computing*, page 103927, 2020.
- Stylianos Moschoglou, Athanasios Papaioannou, Christos Sagonas, Jiankang Deng, Irene Kotsia, and Stefanos Zafeiriou. Agedb: the first manually collected, in-the-wild age database. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops*, pages 51–59, 2017.
- Sivaram Prasad Mudunuri, Soubhik Sanyal, and Soma Biswas. Genlr-net: Deep framework for very low resolution face and object recognition with generalization to unseen categories. In *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*, pages 602–60209. IEEE, 2018.
- Nasrabadi NM et al. Identity-aware deep face hallucination via adversarial face verification. In *IEEE International Conference on Biometrics Theory Applications and Systems*, 2019.
- Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng

- Huang, Andrej Karpathy, Aditya Khosla, Michael Bernstein, et al. Imagenet large scale visual recognition challenge. *International journal of computer vision*, 115(3):211–252, 2015.
- 21 Florian Schroff, Dmitry Kalenichenko, and James Philbin. Facenet: A unified embedding for face recognition and clustering. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 815–823, 2015.
  - 22 Soumyadip Sengupta, Jun-Cheng Chen, Carlos Castillo, Vishal M Patel, Rama Chellappa, and David W Jacobs. Frontal to profile face verification in the wild. In *2016 IEEE Winter Conference on Applications of Computer Vision (WACV)*, pages 1–9. IEEE, 2016.
  - 23 Maneet Singh, Shruti Nagpal, Richa Singh, Mayank Vatsa, and Angshul Majumdar. Magnifyme: Aiding cross resolution face recognition via identity aware synthesis. *arXiv preprint*, 2018. [arXiv:1802.08057](https://arxiv.org/abs/1802.08057).
  - 24 Christian Szegedy, Wei Liu, Yangqing Jia, Pierre Sermanet, Scott Reed, Dragomir Anguelov, Dumitru Erhan, Vincent Vanhoucke, and Andrew Rabinovich. Going deeper with convolutions. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1–9, 2015.
  - 25 Veeru Talreja, Fariborz Taherkhani, Matthew C Valenti, and Nasser M Nasrabadi. Attribute-guided coupled gan for cross-resolution face recognition. *arXiv preprint*, 2019. [arXiv:1908.01790](https://arxiv.org/abs/1908.01790).
  - 26 Su Tang, Shan Zhou, Wenxiong Kang, Qiuxia Wu, and Feiqi Deng. Finger vein verification using a siamese cnn. *IET Biometrics*, 8(5):306–315, 2019.
  - 27 Hao Wang, Yitong Wang, Zheng Zhou, Xing Ji, Dihong Gong, Jingchao Zhou, Zhifeng Li, and Wei Liu. Cosface: Large margin cosine loss for deep face recognition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 5265–5274, 2018.
  - 28 Qiangchang Wang, Tianyi Wu, He Zheng, and Guodong Guo. Hierarchical pyramid diverse attention networks for face recognition. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 8326–8335, 2020.
  - 29 Zhifei Wang, Zhenjiang Miao, QM Jonathan Wu, Yanli Wan, and Zhen Tang. Low-resolution face recognition: a review. *The Visual Computer*, 30(4):359–386, 2014.
  - 30 Yandong Wen, Kaipeng Zhang, Zhifeng Li, and Yu Qiao. A discriminative feature learning approach for deep face recognition. In *European conference on computer vision*, pages 499–515. Springer, 2016.
  - 31 Erfan Zangeneh, Mohammad Rahmati, and Yalda Mohsenzadeh. Low resolution face recognition using a two-branch deep convolutional neural network architecture. *Expert Systems with Applications*, 139:112854, 2020.
  - 32 Dan Zeng, Hu Chen, and Qijun Zhao. Towards resolution invariant face recognition in uncontrolled scenarios. In *2016 International Conference on Biometrics (ICB)*, pages 1–8. IEEE, 2016.
  - 33 Kai Zhang, Wangmeng Zuo, and Lei Zhang. Deep plug-and-play super-resolution for arbitrary blur kernels. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 1671–1681, 2019.
  - 34 Kaipeng Zhang, Zhanpeng Zhang, Chia-Wen Cheng, Winston H Hsu, Yu Qiao, Wei Liu, and Tong Zhang. Super-identity convolutional neural network for face hallucination. In *Proceedings of the European conference on computer vision (ECCV)*, pages 183–198, 2018.
  - 35 Tianyue Zheng and Weihong Deng. Cross-pose lfw: A database for studying cross-pose face recognition in unconstrained environments. *Beijing University of Posts and Telecommunications, Tech. Rep*, 5, 2018.
  - 36 Tianyue Zheng, Weihong Deng, and Jiani Hu. Cross-age lfw: A database for studying cross-age face recognition in unconstrained environments. *arXiv preprint*, 2017. [arXiv:1708.08197](https://arxiv.org/abs/1708.08197).
  - 37 Ruofan Zhou and Sabine Susstrunk. Kernel modeling super-resolution on real low-resolution images. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 2433–2443, 2019.



# Micro- and Macroscopic Road Traffic Analysis using Drone Image Data

Friedrich Kruber ✉ 

Technische Hochschule Ingolstadt, Esplanade 10, Ingolstadt, Germany

Eduardo Sánchez Morales ✉ 


Technische Hochschule Ingolstadt, Esplanade 10, Ingolstadt, Germany

Robin Egolf ✉ 

Technische Hochschule Ingolstadt, Esplanade 10, Ingolstadt, Germany

Jonas Wurst ✉ 

Technische Hochschule Ingolstadt, Esplanade 10, Ingolstadt, Germany

Samarjit Chakraborty ✉ 

University of North Carolina at Chapel Hill (UNC), Department of Computer Science, NC 27599, USA

Michael Botsch ✉ 

Technische Hochschule Ingolstadt, Esplanade 10, Ingolstadt, Germany

## Abstract

The current development in the drone technology, alongside with machine learning based image processing, open new possibilities for various applications. Thus, the market volume is expected to grow rapidly over the next years. The goal of this paper is to demonstrate the capabilities and limitations of drone based image data processing for the purpose of road traffic analysis.

In the first part a method for generating microscopic traffic data is proposed. More precisely, the state of vehicles and the resulting trajectories are estimated. The method is validated by conducting experiments with reference sensors and proofs to achieve precise vehicle state estimation results. It is also shown, how the computational effort can be reduced by incorporating the tracking information

into a neural network. A discussion on current limitations supplements the findings. By collecting a large number of vehicle trajectories, macroscopic statistics, such as traffic flow and density can be obtained from the data. In the second part, a publicly available drone based data set is analyzed to evaluate the suitability for macroscopic traffic modeling. The results show that the method is well suited for gaining detailed information about macroscopic statistics, such as traffic flow dependent time headway or lane change occurrences.

In conclusion, this paper presents methods to exploit the remarkable opportunities of drone based image processing for joint macro- and microscopic traffic analysis.

**2012 ACM Subject Classification** Computing methodologies → Machine learning

**Keywords and Phrases** traffic data analysis, trajectory data, drone image data

**Digital Object Identifier** 10.4230/LITES.8.1.2

**Supplementary Material** *Software (Source Code)*: <https://github.com/fkthi/OpenTrafficMonitoringPlus>

**Acknowledgements** The authors acknowledge the financial support by the Federal Ministry of Education and Research of Germany (BMBF) in the framework of FH-Impuls (project number 03FH7I02IA).

**Received** 2021-05-11 **Accepted** 2022-01-29 **Published** 2022-11-16

**Editor** Samarjit Chakraborty and Qing Rao

**Special Issue** Special Issue on Embedded Systems for Computer Vision

## 1 Introduction

Drones are an excellent example of the ongoing technological development of embedded systems. The commercial drone market is already a multi-billion dollar business. Growth rates of 15 percent per year are forecast for the coming years [13]. The capability of drones to carry payloads make

## 02:2 Drone Image Data For Joint Micro- and Macroscopic Road Traffic Analysis

them versatile tools. They are used to detect biological microorganisms or chemical substances, to inspect technical equipment such as wind turbines and gas pipelines, or to monitor crop growth and measuring biomass, to name a few examples [56]. Most commonly, drones are equipped with cameras. The development of drones as embedded vision systems runs parallel to breakthroughs in computer vision using deep learning methods. This increases the potential for automated analysis of image data.

The present work is specifically dedicated to the application for traffic surveillance and analysis. Traffic is traditionally analyzed either on a global, macroscopic view, or in contrast, on a vehicle based, microscopic view. Section 2 summarizes typical sensor setups and study procedures from established methods.

Section 3 proposes methods for some of the core elements for the data acquisition via drones: image registration, object detection, coordinate transformation and object tracking. The requirements for object detection are high, since every pixel translates into several centimeters on the ground. For example, it is crucial to know which lane a vehicle is located in. Thus, the tolerance is accordingly in the range of a few decimeters or pixels, respectively. In order to map the images to the real world, the pixelized information has to be transformed to real world coordinates. Since a hovering drone is exposed to wind, slight movements have to be compensated with image registration techniques. By tracking vehicles over time, state variables, such as position over time, speed, acceleration and orientation can be obtained. To validate the proposed method of estimating the vehicle's state, experiments with reference sensors are conducted in this work. Lastly in Section 3, a novel approach to reduce the computational load of the object detection is proposed. For this, Kalman Filter based predictions are fed into a neural network as region proposals, which allows to deactivate one part of the neural network temporarily and increases the average throughput. The code for the complete methodology is available at <https://github.com/fkthi/OpenTrafficMonitoringPlus>.

By collecting a large number of vehicle trajectories, macroscopic traffic statistics can be derived as well. In Section 4, a publicly available dataset, acquired by means of drones, is used to validate the approach for macroscopic traffic statistics. The synchronization of the micro- and macroscopic domain can improve traffic modeling, but is difficult to accomplish with established approaches. Now, information from both domains can be captured synchronous by a single camera, mounted on a drone. Questions like how likely are lane changes performed in relation to the traffic flow, how do vehicles distribute on multi-lane roads, or how do drivers adapt distances related to the traffic situation, can be answered with bird's-eye view images from hovering drones.

Possible applications are accordingly diverse. For example, the data can be used to understand risk factors for traffic accidents by analyzing the behavior of traffic participants in intersections. Macroscopic traffic simulations benefit from real traffic measurements too. They are applied in many fields, for example to predict the effects of road network modifications, to optimize traffic signal coordination for green wave traffic, or to improve emergency vehicles travel times [10]. Roadside unit sensors are expensive to plan, install and maintain, while drone recording campaigns can be carried out at every place with low effort, so that investigations can also be conducted off the main roads and at very short notice. Such data supports engineers in making traffic in various situations more predictable and controllable as well.

Other applications target the automotive industry. Trajectory prediction and path planning are two main challenges for automated driving. Here, data is typically collected with test vehicles, equipped with reference sensors. However, the field of view from the vehicle perspective is limited because of occlusions and the range of vision, which negatively affects the understanding of other traffic participants decisions and actions. Images from aerial campaigns do not suffer from occlusion and observe a large area on ground. They allow data collection for many vehicles in parallel and capture how individual objects interact with each other.



## 2 Related work

This section starts with an overview on typical measurement principles for macroscopic data, followed by the microscopic data collection. The third subsection summarizes the current state of traffic surveillance from aerial imagery and provides an overview of public available data sets.

### 2.1 Macroscopic data

Traffic flow theory has been intensively researched for decades, starting with Greenshields publication and the first fundamental diagram in 1933 [18]. The works of [20, 23] provide a broad overview of the literature research of traffic stream characteristics and comparisons of several approaches on how to obtain data. Table 1 summarizes the measurement principles, typical sensors and the obtained measurements. It should be noted that, while some variables cannot be measured with certain methods, they can still be estimated to some extent. For example, single loop detectors cannot estimate speed, but if several of them are placed within distances of some hundred meters, average speed can still be estimated. Around 90 % of fixed sensors are induction loops [9], which are buried in the pavement. Radars and cameras are increasingly becoming established and are constantly being further developed. Floating car data, acquired by cell phones, enables end-user services for estimated travel times and alternative routes. Electronic toll collection is automated with RFID tag equipped vehicles. By measuring the time between consecutive toll readers, the travel times can be estimated as well [9]. The type of variables that can be captured by a fixed camera depends on the mounting location. For example, if they are located on a high building to perceive a large area, they offer the same possibilities as a drone. A major advantage of image-based methods is that they can deliver both, point-based and distance-based variables. Looking at an individual image frame, the distance-based traffic density can be estimated. However, if one considers a sequence of video frames, the number of vehicles crossing a specific location over time yields the traffic flow.

The costs for fixed sensor units are estimated up to USD 20,000, depending on existing structures [15]. Once installed, they deliver a constant data stream and they are less affected by weather conditions compared to drones. In contrast to that, drone data can only be obtained within certain time windows and the data acquisition causes variable costs. Batteries are a bottleneck, but this disadvantage can nowadays be compensated by tethered drone systems, which allow flight durations of several hours. Wind and water resistance, alongside with low-light capabilities of cameras, are still weak points. In return, drone based data acquisition has three key advantages: 1) data can be recorded without the necessity of additional infrastructure and 2) both, point-based and distance-based variables can be obtained, and 3) it enables joint micro- and macroscopic traffic analysis.

### 2.2 Microscopic data

In this work, microscopic data is defined as the estimation of vehicle state variables, such as speed, accelerations and rotation rates. By observing several nearby located vehicles, their interactions can be inferred as well. In contrast to that, the aggregation of data from many vehicles over time or space defines the macroscopic view. While the macroscopic analysis is rather of interest for transportation planners, the microscopic data is targeted towards the automotive industry. Typical data recording procedures from the ego vehicle perspective are:

- Field Operational Tests (FOT),
- Natural Driving Studies (NDS).

■ **Table 1** Measurement principles, available sensors and corresponding traffic variables for macroscopic traffic stream characteristics.

Method	Sub-Method	Measurements						
		Speed	Flow	Density	Occupancy	Travel Time	Headway	Vehicle class
Fixed sensor	single loop		x		x			
	dual loop	x	x		x		x	x
	radar	x	x		x	x	x	x
	camera	x	x	x	x	x	x	x
Drone	hovering	x	x	x	x	x	x	x
Floating cars	GPS / Cell Phones	x				x		
	Transponder (RFID)					x		

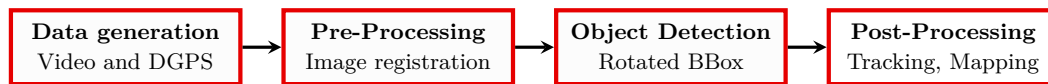
The two procedures differ according to their target [35]. In FOT, the drivers get instructions in order to validate or assess a certain functionality. As an example, the drivers are asked to evaluate a lane keeping assist. In contrast to that, NDS are free of instructions in order to analyze unbiased driving behavior. For example, NDS can be used to examine how fast drivers approach an uncontrolled intersection, or how often they perform lanes changes. Car manufacturers proceed similarly. Test drives can be performed to validate a specific functionality according to given specifications. Alternatively, long term tests can be carried out without specific instructions in order to reveal any sort of unexpected malfunction, especially in the context of system integration. In both cases, FOT and NDS, vehicles can be either equipped with series production sensors, or additionally with reference sensors, such as Lidars and additional cameras.

The scope of in-vehicle testing spans all aspects of driving: perception, planning, and execution. In contrast to that, recordings from external sensors capture only the output of these tasks. Nevertheless, such data is valuable for several purposes, such as developing trajectory planning algorithms, where the drone based recordings function as ground truth data. Another application is the development of behavior models of traffic participants.

### 2.3 Traffic data acquired with drones

The interest in traffic data acquisition with drones is underlined by the increasing number of publications over the past years. Here, related works are divided into object detection methods from aerial imagery, followed by publicly available data sets.

Some works propose architectures for object detection from satellite or airplane images [34, 4, 41], while the present work pursues a precise state estimation from flight altitudes of up to 100 m. Other works focus on detection and tracking from drones or infrastructure cameras [19, 53, 32]. DroNet [32] is a lean implementation of the YOLO network [45], where the number of filters in each layer is reduced. DroNet outputs several frames-per-second (fps) with onboard hardware, but at the cost of lower detection performance and image resolution. The network struggles with variations in flight height and vehicle sizes. It outputs horizontal bounding boxes, which are not favorable for estimating variables such as the vehicle orientation or yaw rate. DroNet is rather implemented for vehicle detection with a moving drone, than for vehicle state estimation. The  $R^3$  network [34] enables the detection of rotated bounding boxes from high altitude images.  $R^3$  is a bounding box detector, while the method in this work uses Mask R-CNN [22], which is an instance segmentation network. Except for [53], no other work focuses on the vehicle tracking and state estimation.



■ **Figure 1** The overall process: From data recording to tracking.

The work in [19] approximates the most to the present work. A test vehicle was equipped with a GPS logger that receives positions and speed. The images were geo-referenced to obtain a fixed frame. The main differences with the present work can be stated as follows: The detection algorithm in [19] compares the differences between two frames, hence identifying moving objects by localizing altered pixel values. This type of detector is prone to errors, e.g., during vehicle standstills, changing light conditions or due to the movement of vegetation, as stated by the authors. The output is a non-rotated bounding box, which fails to estimate the vehicles shape and thereby worsens the state estimation. The images in [19] were processed with a Gaussian blur filter, which is claimed to eliminate high frequency noise. Applying such a filter blurs the edges and is contra productive when applying a neural network detector. Finally, relief displacement was not taken into account, which causes an increasing error with growing distance to the principal point, see Section 3. The authors state a normalized root mean square error of 0.55 m at a flight altitude of 60 m. By the same measure, the error obtained in the present work is much lower with 0.18 m at a flight altitude of 75 m and identical image resolution. Finally, the reference sensor used in [19] has an accuracy of 0.2 m and 0.03 m/s in the position and velocity respectively, while the one used in this work has a position and velocity accuracy of 0.01 m and 0.01 m/s accordingly. A better reference sensor accuracy allows a pixel-accurate comparison, which makes the experiments more relevant.

Regarding the public data sets, two different types are available. The first type provides pre-processed trajectory data [28, 58, 47, 12, 6]. With more trajectory data sets available now, not only vehicles are tracked, but also bicycles and pedestrians among other classes. Also, the types of locations broaden, from highway to urban infrastructure, such as crossings and roundabouts. The dataset in [6] extends the perceptive field throughout a complete city district with a swarm of several drones flying simultaneously.

The other type of data sets provide labeled training data to improve computer vision methods and tracking algorithms. Large data sets can be found in [42, 59]. Despite their size, these are less useful for traffic monitoring and vehicle state estimation, since the drone does not hover but flies without providing the flight meta data of the drone. Additionally, it was found that [59] appears to lack partly label quality, which is a key factor to obtain good results. Other data sets, such as [40, 11] provide additionally flight meta data including the measurements from GPS and a Inertial Measurement Unit (IMU). In these works, the drones fly at low altitudes, thus capturing a smaller area on ground. The scope of these works is rather on visual odometry, Simultaneous Localization and Mapping (SLAM) or autonomous aerial surveillance.

### 3 Drone image based vehicle state estimation

This section builds on our previous work in [30, 50] and describes a method to acquire a highly accurate vehicle state based on computer vision detection. An overview of the main steps is depicted in Figure 1. In the present work, the method is extended by feeding Kalman Filter [24] based proposals into the neural network in order to reduce the computational load.

The section starts with a description of the coordinate systems, followed by the method description and experiments.

### 3.1 Coordinate systems

The coordinate systems used in this work are described in what follows. The vehicles move on the Local Tangent Plane (LTP), where  $x_L$  points east,  $y_L$  north and  $z_L$  upwards, with an arbitrary origin  $o_L$  on the surface of the earth. The Local Car Plane (LCP) is defined according to the ISO 8855 norm, where  $x_C$  points to the hood,  $y_C$  to the left seat,  $z_C$  upwards, with the origin  $o_C$  at the center of sprung mass of the vehicle. For simplification, it is assumed that

- 1) the  $x_C y_C$ -plane is parallel to the  $x_L y_L$ -plane,
- 2) the centre of sprung mass and geometrical centre of the vehicle are identical, and
- 3) the sensor in the vehicle measures in the LCP.

Quantities expressed in the LTP and LCP are given in the International System of Units (SI). The Pixel Coordinate Frame (PCF) is a vertical image projection of the LTP, where  $x_P$  and  $y_P$  represent the axes, with the origin  $o_P$  in one corner of the image. It is assumed that the camera is pointing vertically downwards. Quantities expressed in PCF are given in pixels (px).

In the following, vectors are represented in boldface and matrices in boldface, capital letters.

### 3.2 Data generation setup

The data set was recorded on a test track. This gives certain degrees of freedom regarding arbitrary vehicle trajectories within the image frame and allows the experiments to be repeated with the same setup. The test vehicle was equipped with an Inertial Navigation System (INS)<sup>1</sup>, which serves as a reference sensor for the position, velocity, acceleration, orientation and rotation rate. This INS is equipped with servoaccelerometers, optical gyroscopes and receives Real-Time Kinematic (RTK) correction data. The Correvit sensor<sup>2</sup> estimates the linear velocities ( $v_x, v_y$ ) along the longitudinal and lateral axes of the vehicle by means of an optical grid [21]. According to [16] the sideslip angle of the vehicle is defined as

$$\beta = \arctan\left(\frac{v_y}{v_x}\right). \quad (1)$$

Therefore, the Correvit can serve as the reference sensor for the estimated sideslip angle.

Table 2 depicts the flight altitudes and Ground Sampling Distance (GSD) for the drone<sup>3</sup> in use, Section 3.5.1 details the computation.

■ **Table 2** Total count of frames and GSD per altitude.

Flight altitude	50 m	75 m	100 m
Number of frames	14 532	15 217	24 106
GSD [cm/px]	3.5	5.2	6.9

Generally, for a vertical photograph, the GSD  $S$  is a function of the camera's focal length  $f$  and flight altitude  $H$  above ground:

$$S = \frac{f}{H}. \quad (2)$$

<sup>1</sup> GeneSys ADMA-G-PRO+

<sup>2</sup> Kistler Correvit S-Motion

<sup>3</sup> DJI Phantom 4 Pro V2



■ **Figure 2** Registered image (left) and the raw drone image (right) of a Ground Control Point (GCP) from 1.5 m height. The borders of the left image are clipped due to translation and rotation. The red box depicts the GCP location at the first video frame, which was shot around 30 s beforehand.

Varying altitudes brings flexibility in the trade-off between GSD and the captured area on the ground. For each altitude, from 50 m to 100 m, several videos were recorded. Note, that minor, unavoidable altitude differences during hovering are compensated by the image registration, see Section 3.3. The camera frame-rate  $f_f$  was set to  $f_f = 50$  fps and exposure time was kept constant at  $\frac{1}{400}$  s for all recordings.

### 3.3 Pre-processing

The pre-processing consists generally of two parts: The camera calibration and the image registration. According to the manufacturer of the drone, the camera is shipped calibrated, so this step is skipped. The image registration is performed to overlay the sequential video frames over the first one to ensure a fixed image frame. The registration implemented in this work is composed of a correction of the orientation, translation, and scaling of the image. Figure 2 depicts an example of the registration result. This process involves three steps in order to find correspondences between two images: a feature detector, a descriptor and finally the feature matching. The goal of the detector is to find identical points of interest under varying viewing conditions. The descriptor is a feature vector, which describes the local area around the point of interest. To match the points between two images, the distances between the feature vectors are computed. If the distance fulfills a certain criterion, e. g., a nearest neighbor ratio matching strategy, a matching point on two images is found. The matches are then fed into the MLESAC algorithm [54] to eliminate outliers. Lastly, a randomly selected subset of the remaining matching points is used for the image scaling, rotation and translation. The scenery recorded should offer some distinguishable, static features to ensure a robust image registration. Since all consecutive video frames are compared to the very first frame, a confusion between static and moving objects can be avoided. In this work, two competitive algorithms are applied and compared according to their execution time: SURF [7] and ORB [49], based on their openCV implementation. Additional information can be obtained from the survey [27].

### 3.4 Object detection

In addition to the object detection, an estimation of the dimensions and orientation of the vehicles is required for many applications. Semantic segmentation networks are suitable for this purpose, since they detect objects in random shapes, based on a pixelwise prediction. From these shapes rotated rectangles can be derived, which serve as bounding boxes for the vehicles. In particular, networks of the subgroup of instance segmentation are advantageous, since these networks output directly object wise instances, instead of a pixelwise class prediction over the complete image. For





■ **Figure 3** Detection examples: The left column depicts examples from the experiments performed on the test track. Other images are taken on roads, partly from publicly available sources [59, 44]. The masks, depicted with random colors, are used to compute the location, size and orientation of the smallest rectangle containing all mask pixels of the detected object. Note, the non-rotated bounding boxes would be the final output of typical object detectors without the segmentations masks.

this work, the Mask R-CNN network [22] is chosen for its strong detection performance for traffic surveillance with drone images. Mask R-CNN extends Faster R-CNN [46] by adding a parallel, Fully Convolutional Network [39] branch for instance segmentation, next to the classification and bounding box regression from Faster R-CNN. The network is a so called two stage detector: In the first stage, feature maps generated by a backbone network are fed into a Region Proposal Network (RPN), which outputs Regions of Interest (RoI). In the second stage, the predictions are performed within the RoI Heads. One head predicts classification and regression, the second head provides segmentation masks. These masks are used to compute the location, size and orientation of the smallest rectangle containing all mask pixels of the detected object, in our case a vehicle. The combination of the RPN with a Feature Pyramid Network (FPN) [37], both part of Mask R-CNN, achieve strong detection results for rather small objects and additionally for objects of different scales, which result from varying flight altitudes, as well as the varying objects sizes from a small car up to a large truck, see Figure 3 for examples.

The network is pre-trained on the Common Objects in Context (COCO) data set [38]. To predict vehicles from the top view, transfer learning has been applied with an own, manually labeled data set. Further details, along to the extension with Kalman Proposals are presented in Section 3.9. Implementation details are provided in our repository. Figure 3 depicts some detection examples. The left column depicts examples from the experiments performed on the test track (Section 3.7). Other images are taken on roads, partly from publicly available sources [59, 44]. Note, the non-rotated bounding boxes in Figure 3 would be the final output of typical object detectors without the segmentations masks.

### 3.5 Post-processing steps

To complete the process for a single image, two more steps are performed. First, the output from the neural network, given in PCF, has to be mapped to the LTP. The mapping in this work is applied in order to perform the experiments with the reference sensors. Another use-case for the mapping is the translation of the pixelized information onto a road map. In the second step, measures are taken to reduce errors induced by the relief displacement.

#### 3.5.1 PCF Mapping

A comparison to the reference sensors requires the mapping of the PCF to the LTP. For this, GCPs are placed on the  $x_L y_L$ -plane, in such a way that they are visible on the image. The  $i$ -th GCP is defined in LTP as  $\mathbf{g}_{i,L} = [x_{i,L} \ y_{i,L}]^T$ , and in PCF as  $\mathbf{g}_{i,P} = [x_{i,P} \ y_{i,P}]^T$ . The GSD  $S$  is calculated from two GCPs by

$$S = \frac{|\mathbf{g}_{i+1,L} - \mathbf{g}_{i,L}|}{|\mathbf{g}_{i+1,P} - \mathbf{g}_{i,P}|}. \quad (3)$$

The  $i$ -th GCP can then be expressed in meters by

$$\tilde{\mathbf{g}}_i = \mathbf{g}_{i,P} \cdot S = [\tilde{x}_i \quad \tilde{y}_i]^T. \quad (4)$$

The orientation offset  $\delta$  from the LTP to the PCF is calculated as

$$\delta = \theta_i - \theta_{i,L}, \text{ with} \quad (5)$$

$$\theta_i = \text{atan2}(\tilde{y}_{i+1} - \tilde{y}_i, \tilde{x}_{i+1} - \tilde{x}_i), \quad (6)$$

$\theta_{i,L}$  is calculated by analogy. The GCP  $\tilde{\mathbf{g}}_i$  is then rotated as follows

$$\hat{\mathbf{g}}_i = \mathbf{R}_r(\delta)^T \tilde{\mathbf{g}}_i, \quad (7)$$

where  $\mathbf{R}_r(\cdot)$  is a 2D rotation matrix. The linear offsets from the LTP to the PCF are calculated by  $\Delta = \hat{\mathbf{g}}_i - \mathbf{g}_{i,L}$ . Finally, a pixel  $\mathbf{p}_P = [x_P \quad y_P]^T$  on the PCF can be mapped to the LTP by

$$\mathbf{p}_P^L = \left( \mathbf{R}_r(\delta)^T (\mathbf{p}_P \cdot S) \right) - \Delta. \quad (8)$$

The next stage is to semantically define the four bounding box corners. It is assumed that the box covers the complete shape of the vehicle. The  $i$ -th corner of the bounding box is defined in PCF as  $\mathbf{b}_i = [x_{b,i,P} \quad y_{b,i,P}]^T$ , and the bounding box is defined in PCF as

$$\mathbf{B}_P = [\mathbf{b}_1 \quad \mathbf{b}_2 \quad \mathbf{b}_3 \quad \mathbf{b}_4]. \quad (9)$$

The corners of the bounding box are mapped to the LTP as shown in Eq. (8) to obtain  $\mathbf{B}_P^L$ . The geometric centre of the vehicle  $\mathbf{o}_{\text{veh}}$  is calculated by

$$\mathbf{o}_{\text{veh}} = \begin{bmatrix} \frac{\max(\mathbf{B}_{P^L}^L 1,i) + \min(\mathbf{B}_{P^L}^L 1,i)}{2} \\ \frac{\max(\mathbf{B}_{P^L}^L 2,i) + \min(\mathbf{B}_{P^L}^L 2,i)}{2} \end{bmatrix}, \quad (10)$$

for  $i = 1, \dots, 4$ . The dimensions of the detected vehicle are calculated next. Let

$$\|\mathbf{b}_2 - \mathbf{b}_1\| < \|\mathbf{b}_3 - \mathbf{b}_1\| < \|\mathbf{b}_4 - \mathbf{b}_1\|, \quad (11)$$

then  $\hat{w} = S \cdot \|\mathbf{b}_2 - \mathbf{b}_1\|$  and  $\hat{l} = S \cdot \|\mathbf{b}_3 - \mathbf{b}_1\|$  are the estimated width  $\hat{w}$  and length  $\hat{l}$  of the vehicle in meters.

Knowing this, the orientation  $\psi_{\text{veh}}^L$  of the vehicle is given by

$$\psi_{\text{veh}}^L = \text{atan2}(y_{j,P}^L - y_{1,P}^L, x_{j,P}^L - x_{1,P}^L), \quad (12)$$

where  $j$  is the element of  $\mathbf{B}_P$  associated with the length  $\hat{l}$  of the vehicle. The measurement vector for the Kalman Filter is then defined as

$$\mathbf{z}_{\text{in}} = [\mathbf{o}_{\text{veh}}^T \quad \psi_{\text{veh}}^L]^T. \quad (13)$$

## 02:10 Drone Image Data For Joint Micro- and Macroscopic Road Traffic Analysis

The estimation of the measurement noise is detailed in what follows. As described above, the GSD and the orientation offset can be estimated from one pair of GCPs as long as they are visible on the PCF. By using the same method, if  $n \geq 2$  GCPs are visible on the PCF, then the number of values  $c$  that can be calculated for the GSD and for the orientation offset can be calculated as

$$c = \frac{n!}{2!(n-2)!}, \quad (14)$$

where  $n$  is the number of GCPs. An arithmetic mean  $\bar{S}$  for the GSD can then be calculated as

$$\bar{S} = \frac{\sum_{i=1}^c S_i}{c}, \quad (15)$$

where  $S_i$  is the GSD estimated with the  $i$ -th pair of GCPs. Likewise, an arithmetic mean  $\bar{\delta}$  for the orientation offset can be calculated as

$$\bar{\delta} = \frac{\sum_{i=1}^c \delta_i}{c}, \quad (16)$$

where  $\delta_i$  is the orientation offset estimated with the  $i$ -th pair of GCPs. The measurement error  $\zeta_{o,veh}$  for the position and the measurement error  $\zeta_{\psi,veh}$  for the orientation can then be calculated by

$$\zeta_{o,veh} = \max(S_i - \bar{S}), \text{ with } i \in \{1, \dots, c\}, \quad (17)$$

$$\zeta_{\psi,veh} = \max(\delta_i - \bar{\delta}), \text{ with } i \in \{1, \dots, c\}. \quad (18)$$

The measurement noise for the Kalman Filter is then defined as the diagonal matrix  $\zeta_z$

$$\zeta_z = \text{diag}(\zeta_{o,veh}, \zeta_{o,veh}, \zeta_{\psi,veh}). \quad (19)$$

### 3.5.2 Relief displacement

Photographs yield a perspective projection. A variation in the elevation of an object results in a different scale and a displacement of the object. An increase in the elevation of an object causes the position of the object's feature to be displaced radially outwards from the principal point  $O_c$  [36].

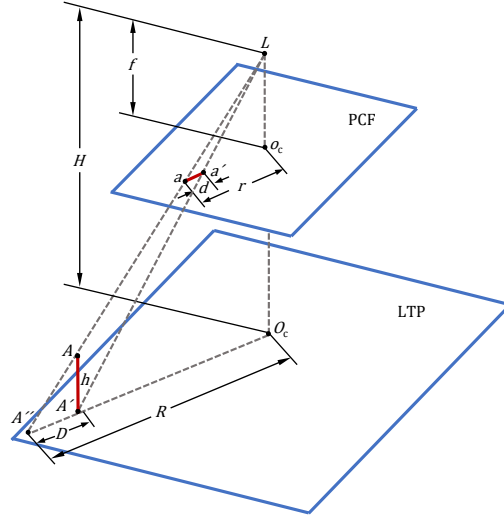
Assuming a vertical camera angle, the displacement can be computed from the similar triangles  $LO_cA''$  and  $AA'A''$ , according to Figure 4:

$$\frac{D}{h} = \frac{R}{H}, \quad \frac{d}{h} = \frac{r}{H}, \quad (20)$$

where the second equation is expressed in GSD, with  $d$  defining the relief displacement and  $r$  the radial distance between  $o_c$  and the displaced point  $a$  in PCF.  $D$  defines the equivalent distance of  $d$ , projected on the ground,  $R$  the radial distance from  $O_c$ ,  $H$  defining the flight altitude and  $h$  being the object height in LTP.  $L$  is the camera lens exposure station, where light rays from the object intersect before being imaged at the cameras' sensor. The relief displacement decreases with an increasing hovering altitude and is zero at  $O_c$ .

According to Eq. 20, the bounding box has to be shifted radially. Two approaches are considered: The first one requires knowledge of the vehicle sizes, and the second one is an approximation for unknown vehicle dimensions. Since the training is performed to detect the complete vehicle body, the corner closest to  $O_c$  can be identified as the bottom of the vehicle body. So the height of this point is equal to the ground clearance. Knowing the height of this corner, its displacement is corrected as described in what follows.





■ **Figure 4** Geometry of the relief displacement, adapted from [36]. The red bar depicts an object of height  $h$ . Due to the perspective projection and  $R > 0$ , the top of the bar  $A$  is displaced on the photo compared to the bottom  $A'$ . The relief displacement  $d$  is the distance between the corresponding points  $a$  and  $a'$  in the PCF.

Defining the horizontal and vertical resolution of the image as  $r_x$  and  $r_y$ , the coordinates in PCF of  $\mathbf{b}_i$  w.r.t. the image center are given by

$$\begin{bmatrix} x_{b,i,\text{img}} \\ y_{b,i,\text{img}} \end{bmatrix} = \begin{bmatrix} x_{b,i,P} - \frac{r_x}{2} \\ y_{b,i,P} - \frac{r_y}{2} \end{bmatrix}. \quad (21)$$

The shift  $\Delta_{x,P}$  along the  $x_P$  axis is calculated on the PCF by

$$\Delta_{x,P} = \frac{x_{b,i,\text{img}} \cdot h_{b,i,L}}{H}, \quad (22)$$

where  $h_{b,i,L}$  is the height of the  $i$ -th corner on the LTP. The shift for  $\Delta_{y,P}$  is computed by analogy along the  $y_P$  axis. The shifted coordinates  $\mathbf{b}_{i,\text{shift}}$  of  $\mathbf{b}_i$  are then given by

$$\mathbf{b}_{i,\text{shift}} = \mathbf{b}_i - [\Delta_{x,P} \quad \Delta_{y,P}]^T. \quad (23)$$

Let  $w$  be the width and  $l$  the known length of the vehicle and  $\mathbf{b}_1$  be the closest corner to the image centre. Then,  $\mathbf{b}_1$  is used for scaling  $\mathbf{b}_2$  and  $\mathbf{b}_3$  as follows

$$\mathbf{b}_{w,\text{scaled}} = \left( \frac{w}{\hat{w}} \cdot (\mathbf{b}_2 - \mathbf{b}_1) \right) + \mathbf{b}_1, \text{ and} \quad (24)$$

$$\mathbf{b}_{l,\text{scaled}} = \left( \frac{l}{\hat{l}} \cdot (\mathbf{b}_3 - \mathbf{b}_1) \right) + \mathbf{b}_1, \quad (25)$$

where  $w$  is the element of  $\mathbf{B}_P$  associated with  $\|\mathbf{b}_2 - \mathbf{b}_1\|$  and  $l$  associated with  $\|\mathbf{b}_3 - \mathbf{b}_1\|$ , respectively. The shifted centre of the vehicle can then be calculated by

$$\mathbf{o}_{\text{veh,shift}} = \frac{\mathbf{b}_{w,\text{scaled}} + \mathbf{b}_{l,\text{scaled}}}{2}. \quad (26)$$

When gathering data on public roads, the vehicle dimensions are unknown and cannot be estimated with a mono camera. An approximation for the displacement can be performed by assuming that two sides of the bounding box closest to  $o_c$ , are collinear to the lowest part of the vehicle chassis. The ground clearance can be approximated as 15 cm for passenger cars [55]. The remaining two sides can usually be referred to as the vehicle body shoulders, which usually protrude further than the roof of the vehicle. The shoulders height is roughly half of the vehicle height and can be approximated with 75 cm for passenger cars. Then all four corners can be shifted following Eq. (22). Although this is only a coarse approximation, the overall error is reduced when compared to the initial situation of neglecting the displacement.

### 3.6 Tracking and state estimation

To enable the object tracking, the detections must be associated across frames in a sequence of images. The association procedure in this paper follows the computationally efficient SORT algorithm [8]. In contrast to batch tracking methods, this algorithm solely requires information from the previous and current frame. It is therefore suitable for real-time applications and endless video recording or streaming. In principle, an Intersection-over-Union (IoU) distance is computed for all detected vehicle shapes from two consecutive frames and stored in a cost matrix. The assignment is computed optimally using the Hungarian algorithm [31]. If no detection is associated to a vehicle no corresponding measurement vector can be generated. However, its state is continuously predicted without measurement variables using the Kalman Filter. After a defined number of frames without association, the vehicle track is withdrawn. The IoU distance allows implicitly short-term occlusions.

Having the measurement vector from Equation (13), the state vector from Equation (27) and measurement noise from Equation (19) assigned to an object, the next step is to estimate the vehicle state using the Kalman Filter as described in [24]. The Kalman Filter also allows to estimate state variables that are not part of the measurement vector by allowing the system noise to propagate. The specifics applicable to this work are described in the following.

The used state vector is defined as

$$\mathbf{x} = [x_{\text{car}}, y_{\text{car}}, v_{x,\text{car}}, v_{y,\text{car}}, a_{x,\text{car}}, a_{y,\text{car}}, \psi_{\text{car}}, \dot{\psi}_{\text{car}}]^T, \quad (27)$$

where  $x_{\text{car}}$  and  $y_{\text{car}}$  are the (x,y) coordinates of  $\mathbf{o}_{\text{car}}$  in LTP,  $v_{x,\text{car}}$  and  $v_{y,\text{car}}$  are the velocities of  $\mathbf{o}_{\text{car}}$  along the  $x_L$  and  $y_L$  axes,  $a_{x,\text{car}}$  and  $a_{y,\text{car}}$  are the accelerations of  $\mathbf{o}_{\text{car}}$  along the  $x_L$  and  $y_L$  axes,  $\psi_{\text{car}}$  is the angle from  $x_L$  to  $x_C$  in LTP, and  $\dot{\psi}_{\text{car}}$  is the yaw rate around  $z_C$ . In this work a linear motion model is used.

Since the course over ground  $\theta_{\text{cog}}$  in LTP of the vehicle is defined as

$$\theta_{\text{cog}} = \text{atan2}(v_{y,\text{car}}, v_{x,\text{car}}), \quad (28)$$

the sideslip  $\beta$  of the car can then be calculated by

$$\beta = \theta_{\text{cog}} - \psi_{\text{car}}. \quad (29)$$

Detailed information about vehicle dynamics and non-tractive driving can be found in [3] and [52].

In this work, the sideslip angle is estimated by means of a Linear Kalman Filter and Equation (29). This produces better results than using an Extended Kalman Filter. This is explained by the fact the sideslip angle is not part of the measurement vector. Using an Extended Kalman Filter would imply the estimation of the sideslip angle by noise propagation, whereas Equation (29) allows a direct calculation.

### 3.7 Experiments

This section details the experiments carried out on the test track for the validation of the proposed methods. The errors are defined as the deviation from the reference sensors to the estimated states. Figure 5 a) depicts the cumulative error curves for the position estimation before applying the Kalman Filter, so that the results represent unfiltered detections. A spiral template trajectory is driven to obtain different vehicle poses and to cover a large are of the image. The test vehicle is then equipped with a driving robot and a Satellite Navigation (SatNav) system that receives RTK correction data. This ensures an identical reproduction of the trajectory for all experiments, and a centimeter-accurate vehicle localization. No markers are placed on the vehicle to approach real-testing conditions on public roads.

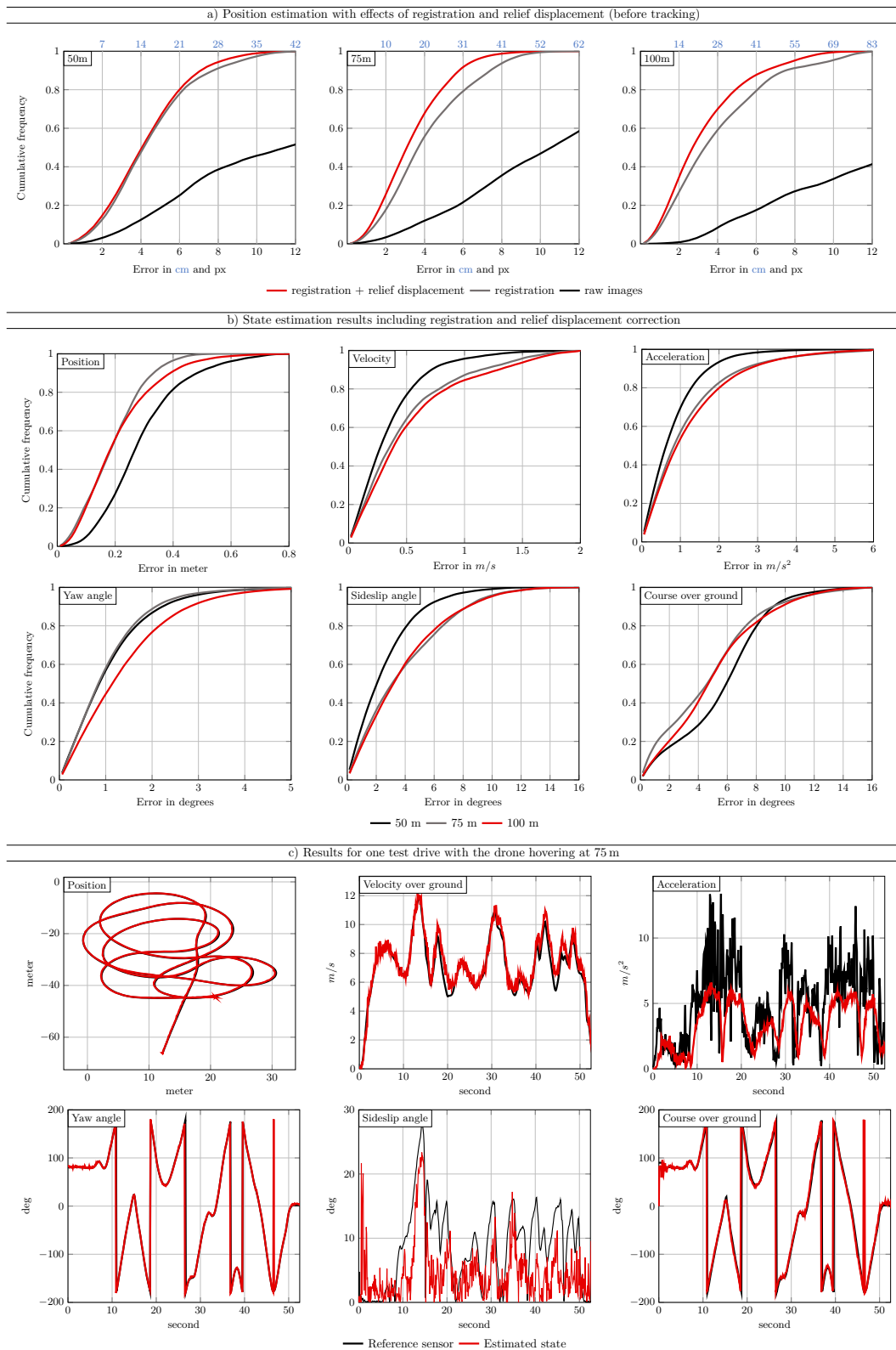
Depicted are curves for each flight altitude and the three main processing steps, where **—** depicts results for non-registered images, **—** for registered images, and **—** for registered images with corrected relief displacement. Image registration is the key to obtain reasonable results. The correction of the relief displacement improves the results by 0.8 px on a weighted average<sup>4</sup>. Note, that the impact of the relief displacement is dependent on the distance  $R$  of the vehicle to the image center  $O_c$ . Hence, data sets recording vehicles at the image border benefit more. The mean position error is 0.2 m and 0.14 m for a flight altitude of 100 m and 50 m, respectively. In terms of pixels, the errors are comparable for all flight heights. Around 90 % of all frames have an error of 7 px or less.

Figure 5 b) depicts the cumulative error curves for all estimated state variables, where **—** depicts cumulative error plots for 50 m, **—** for 75 m and **—** for 100 m hovering altitude, respectively. To generate the data from Figure 5 b) and c), which includes the tracking and Kalman Filter, the vehicle is equipped with the reference sensors and is driven in a random manner on a test track. No specific maneuver is driven in order to avoid tuning Kalman Filter parameters for a specific trajectory or specific driving style. The test drives include standstill, walking velocity, high acceleration, hard braking, tractive and non-tractive driving (drifting). The results show that, once steps are taken to minimize errors, the precision of the estimated vehicle state is comparable to the precision of consumer-grade sensors, such as silicon-based INSS or SatNav receivers with no correction data. This with the advantage of being able to record information for various traffic participants with a single drone. Also, the precision of the estimated sideslip angle allows to differentiate between tractive and non-tractive driving [3]. As part of the experiment strategy, the vehicle was forced to perform drift maneuvers, so that the side slip angle reached high values of up to 28°. The sideslip angle is a relevant state variable to determine the vehicle stability.

Figure 5 c) depicts the estimated state variables (**—**) against the reference sensors (**—**) for one test drive, which includes full-throttle acceleration, hard braking and sudden steering. The estimated position, course over ground and yaw angle for this trial have a mean error of 0.19 m, 4.7° and 1.0° respectively. This precision is equivalent to that of consumer-grade INSS. Subfigure c) also shows that the estimated velocity is affected by a dampening effect and by a time delay. Both are caused by the Kalman gains. A test-specific tuning of the gains could help to reduce the velocity error for this trial, but would increase the error for other tests with lower vehicle dynamics. The deviation of the estimated sideslip is caused mainly by the velocity error. This is because the sideslip angle is estimated using the course over ground, which is calculated from the velocity. During this test, a sideslip angle of 28° is reached, which clearly indicates that the vehicle is drifting. The error in the acceleration is explained by two facts: First, the estimated acceleration is calculated by system noise propagation, so it is low-pass filtered. Second, the reference acceleration, that is measured by the INS, includes vibrations from the drivetrain, the tires and the suspension, as well as from the pitch and roll of the vehicle.

<sup>4</sup> weighted by the number of frames per height

## 02:14 Drone Image Data For Joint Micro- and Macroscopic Road Traffic Analysis



■ **Figure 5** Depicted are: a) Cumulative error curves for the position estimation with the effect of registration and relief displacement before applying the KF, b) Cumulative error curves for all estimated state variables after the KF, c) Estimated states variables against the reference sensor for one test drive with high longitudinal and lateral dynamics.

A summary of the experimental results leads to the following conclusions:

- 1) A robust image registration is crucial for a good performance. If the effects of the relief displacement are neglected, larger errors are present in the estimated vehicle state. This error increases as the hovering altitude decreases and as the objects are farther away from the nadir point.
- 2) Considering the pixelwise results, similar performance is observed for all three altitudes, which proves that the traffic data can be obtained at different flight heights by a single Mask R-CNN network. This advantage can also be helpful for detecting other object classes.
- 3) As a consequence of the GSD, the best results in metric units are achieved at lower altitudes. Alternatively, in order to capture a larger surface area, one can raise the drone to higher altitudes, increase the resolution and crop the image if necessary.
- 4) Regarding real-world applications, the vehicle can be associated to a lane with a precise velocity and orientation estimation, and it can be identified whether the vehicle is drifting or not.

To identify the root causes of the deviations from ground truth, the next section analyzes important sources of errors.

### 3.8 Limitations and sources of error

The methodology involves several processing steps with associated error sources. The main sources of errors are listed in what follows:

- pixel ambiguity and blurring effect,
- relief displacement,
- sensor synchronization.

The blurring effect that can be appreciated on the images can be caused by image compression, camera optics or light propagation. It is precisely this blurring effect that prevents to unambiguously associate a point to a GCP, or a pixel to the vehicle during the labeling of training data or object detection. A pixel ambiguity  $\zeta \pm 1$  px in both, the  $x_P$  and  $y_P$  axes is not uncommon. Since the detection error of  $\pm 1$  px per axis is transferred to the bounding box, then  $\mathbf{b}_i$  could have an error of  $\sqrt{2}$  px.

The effect on the PCF to LTP mapping is shown next. Similarly, the effect of pixel ambiguity can be applied to the image registration process, when pixel-pairs of two images do not match exactly. For a GCP, the  $\mathbf{g}_{i,P}$  is rewritten as

$$\mathbf{g}_{i,P} = [x_{i,P} \quad y_{i,P}]^T \pm \zeta. \quad (30)$$

This association ambiguity has an effect on all three parameters that map the PCF to the LTP. The case of the spatial resolution is analyzed first.

Considering a pixel ambiguity of  $\zeta \pm 1$  px per axis and squared pixels, the distance between the true and associated positions of a GCP on the PCF can be of  $\sqrt{2}$  px. This mis-association causes an error on the spatial resolution. The similarity in percentage  $\eta_S$  between the seen and the true values of the spatial resolution is calculated by

$$\eta_S = \frac{|\mathbf{g}_{i+1,P} - \mathbf{g}_{i,P}|}{|\mathbf{g}_{i+1,P} - \mathbf{g}_{i,P}| + 2 \cdot \sqrt{2}\zeta^2}. \quad (31)$$

The Equation (4) is then rewritten as

$$\mathbf{g}'_i = \mathbf{g}_{i,P} \cdot S \cdot \eta_S = [x'_i \quad y'_i]^T. \quad (32)$$

From the Equations (31) and (32), it can be deduced that the effect of  $\eta_f$  increases as  $\mathbf{g}_{i,P} \rightarrow \mathbf{g}_{i+1,P}$ . In a scenario for a Full-HD image, where both GCPs are placed on opposite diagonal corners of the picture, the similarity can drop to 99.87%. For example, if the true value of  $|\mathbf{g}_{i+1,L} - \mathbf{g}_{i,L}|$  is 100 m, a similarity of 99.87% will rescale it as 99.87 m, meaning a 0.13 m difference.

Next, the effect on the orientation offset is analyzed. This is done in pixels to decouple errors caused by  $\eta_s$ . To consider the pixel ambiguity, the Equation (6) is rewritten as

$$\xi_{\theta_{\text{image}}} = \text{atan2}((\tilde{y}_{i+1} - \tilde{y}_i) \pm 2\zeta, (\tilde{x}_{i+1} - \tilde{x}_i) \pm 2\zeta), \quad (33)$$

where  $\zeta$  is multiplied by two because  $\xi_{\theta_{\text{image}}}$  is calculated using two GCPs. Similar as with  $\eta_s$ , the effect of the pixel ambiguity increases as  $\mathbf{g}_{i,P} \rightarrow \mathbf{g}_{i+1,P}$ . In a scenario for a Full-HD image, where the GCPs are in opposing diagonal corners of the image, then  $\eta_{\xi_{\theta_{\text{image}}}}$  could reach  $0.07^\circ$ .

The orientation error affects the rotation step of the PCF to LTP mapping. So, the effect of orientation error increases as the points to map are farther from the rotation axis. For the  $i$ -th corner of the bounding box, the mapping error  $\eta_{b_i}$  due to the orientation error is given by

$$\eta_{b_i} = \left| \mathbf{R} \left( \eta_{\xi_{\theta_{\text{image}}}} \right) \mathbf{b}_i - \mathbf{b}_i \right| \cdot S. \quad (34)$$

For example, if the drone records a Full-HD video while hovering at 50 m,  $\mathbf{b}_i = [1920 \quad 1080]^\text{T}$  and  $\eta_{\xi_{\theta_{\text{image}}}} = 0.07^\circ$ , then  $\eta_{b_i} \approx 0.08$  m.

The error propagation causes a deviation on the linear offsets as well. The linear offset error  $\eta_{\xi_d}$  due to the orientation and scaling errors is expressed by

$$\eta_{\xi_d} = \left( \left( \mathbf{R} \left( \eta_{\xi_{\theta_{\text{image}}}} \right)^\text{T} (\mathbf{g}_{i,P} \cdot \eta_s) \right) - \mathbf{g}_{i,P} \right) \cdot S. \quad (35)$$

In a scenario where the drone records a Full-HD video while hovering at 50 m,  $\mathbf{g}_{i,P} = [1920 \quad 1080]^\text{T}$ , a similarity of 99.87% and orientation error of  $0.07^\circ$ , then  $\eta_{\xi_d} \approx [-0.13 \text{ m} \quad -0.03 \text{ m}]^\text{T}$ .

From the previous, it is deduced that the best way to minimize errors caused by the PCF to LTP mapping, is to locate the GCPs as far from each other as possible. Also, since the direction of the pixel ambiguity is not deterministic, the errors can be compensated by using different combinations of various GCPs.

Next, the effect of the relief displacement is analyzed. The maximum positioning error  $\eta_r$ , when assuming that the true centroid of the vehicle corresponds to the seen centroid yields:

$$\eta_r = \frac{\sqrt{x_{b,i,\text{img}}^2 + y_{b,i,\text{img}}^2} \cdot S \cdot (h - h_c)}{2 \cdot H}. \quad (36)$$

For example, if the drone hovers at 50 m, the vehicle height is  $h = 1.4$  m, the ground clearance is  $h_c = 0.15$  m and  $\mathbf{b}_i$  is on one corner of the image, then  $\eta_r \approx 0.48$  m. From the discussion above it follows that the best way to minimize positioning errors due to relief displacement, is to correct it only for the corner nearest to the centre of the image, and to re-scale the bounding box.

Another relevant source of error is the synchronization of the sensors. This error is only relevant in a multi-sensor setup as used in this work for the experiments, which is usually not required for real world applications. In this work, the synchronization is performed by using the in-built PPS-LED light of a SatNav receiver as reference. The rising edge of the PPS pulse indicates the start of every second. This rising edge is used as a trigger for lighting up a LED that stays on for a determined period of time so that the light can be seen on the image to be processed. Since this LED is part of the SatNav module, the latency between the PPS reception and the LED lighting up is negligible. The start of each second can be known with frame-accuracy by recording this

LED. The limitation of this technique is that the videos are a series of static pictures. Hence, it is not known if the LED lights up when the shutter is closed, creating a synchronization error. The maximum synchronization error  $\eta_\tau$  is given by

$$\eta_\tau = \frac{1}{f_i}. \quad (37)$$

In this work, 50 fps are used, so that  $\eta_\tau \leq 0.02$  s. The positioning error  $\eta_{\text{pos}}$  caused by  $\eta_\tau$  is given by

$$\eta_{\text{pos}} = \sqrt{v_{x,\text{car}}^2 + v_{y,\text{car}}^2} \cdot \eta_\tau. \quad (38)$$

As an example with a vehicle moving with 50 km/h and a drone hovering at 100 m recording a video with 50 fps, then  $\eta_{\text{pos}} \leq 0.25$  m. It can be deduced from what is discussed above that synchronization errors, even in the millisecond order, have a significant influence.

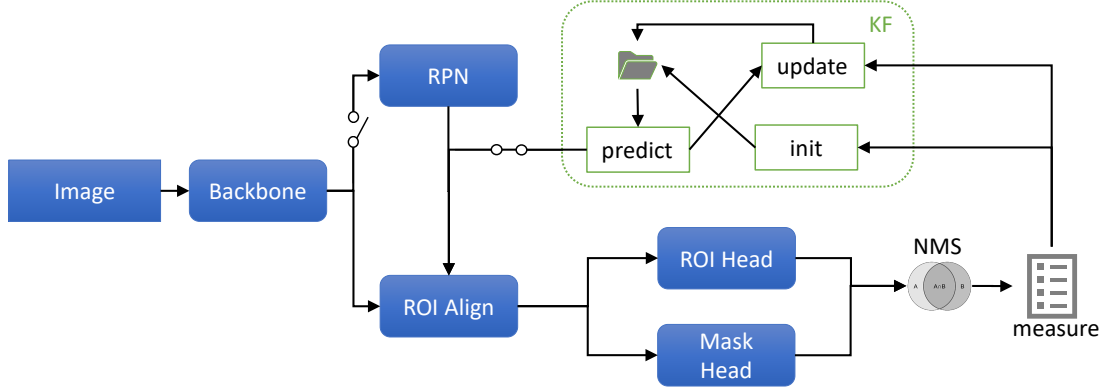
### 3.9 Extension with Kalman Proposals

Common object detection methods are designed for standstill images without spatiotemporal correlation. As a result, a massive amount of anchors are generated throughout the complete image. In the case of Mask R-CNN, these proposals are generated in the Region Proposal Network (RPN). The proposals are ranked by confidence and a predefined number of top ranked proposals is fed into the second stage for the computation of the classification, regression and mask output.

Image sequences are highly correlated, especially when traffic is recorded from a bird's-eye view. Several works propose architectures, which integrate the tracking task into a neural network. Usually, these papers deal with the challenge of the visual appearances changes in natural captured images, i. e. images from a human perspective [57, 26, 25, 5, 17]. To combine detection and tracking in a neural network is appealing. However, the approaches have some of the following drawbacks: 1) The methods require a batch-wise computation, which means they process a complete video sequence at once. The batch-wise computation makes them unsuitable for online tracking, which is required in real time applications such as surveillance video streams. 2) Visual tracking requires storing the information over consecutive frames, which limits the sequences length input.

Changing appearances are not a particular problem for detecting vehicles from bird's-eye view images. Therefore, in this paper an alternative approach is introduced, which efficiently utilizes the existing information from previous frames to accelerate the detection by coupling the Kalman Filter predictions into the neural network. Compared to the related works above, this approach is forward capable, i. e. one is not restricted to run through a complete video sequence at once. The Kalman proposal method can be added to any detector which predicts the output based on region proposals. The Kalman predicted proposals are guided into the second stage of the detector and replace the RPN network for a defined time window. Since only few Kalman Proposals are sufficient for accomplishing the task, less computational resources are required when compared to the typical brute-force proposal generation. To match the expected input for the ROI heads, the Kalman Proposals are fed in as non-rotated bounding boxes.

For initializing the tracks, the RPN is still required and switched on for a certain amount of frames. Then it is turned off and only Kalman Proposals are fed into the second stage of Mask R-CNN. The region proposals for the next video frames are estimated based on the Kalman Filter predicted position of the vehicle and its estimated size. In order to feed these regions to the Mask R-CNN network, they are transformed from the LTP back to the PCF. Afterwards, the same cycle is repeated for detecting new objects. Figure 6 illustrates the changes applied to a Mask R-CNN network, with Kalman Predictions active and the RPN deactivated.



■ **Figure 6** Kalman-Proposals, as part of the assignment and tracking task, are fed into a Mask R-CNN network. The RPN can be switched on and off depending on a time-based event. When the RPN is switched off, only Kalman Predictions are fed into the second stage of Mask R-CNN. The output of the Mask Head functions as measurements induced to the tracking task. The final output is acquired from the KF objects, depicted with a folder symbol.

■ **Table 3** Performance results for Kalman Proposals: On the left side the RPN is turned on for a varying number of consecutive frames and turned off for  $\gamma = 23$  frames. On the right side the RPN is turned on for 10 frames and turned off for a varying number of frames, where Kalman Proposals are fed in instead. True Positives are depicted below ( $\checkmark$ ), False Negatives below ( $\times$ ) and hit rates below  $\frac{\checkmark}{\checkmark + \times}$ .

RPN on	RPN off	$\checkmark$	$\times$	$\frac{\checkmark}{\checkmark + \times}$	RPN on	RPN off	$\checkmark$	$\times$	$\frac{\checkmark}{\checkmark + \times}$
3	23	52544	396	99.1%	10	10	52808	132	99.7%
5	23	52574	366	99.2%	10	23	52616	324	99.3%
10	23	52616	324	99.3%	10	50	51973	967	98.1%
20	23	52657	283	99.4%	10	75	51143	1797	96.5%

In order to evaluate the Kalman Proposals, five video sequences with totally 10 000 frames are analyzed. The videos were recorded at a public roundabout location. When the vehicles enter or leave the image, they are not fully visible, which results in a shifted centroid  $\mathbf{o}_{\text{veh}}$ . The centroid is used for tracking, as described in Section 3.6. In other words, during entering or exiting the image frame, the vehicles' state estimations are not valid. Therefore, the tracking generally only starts at a minimum vehicle length ratio  $\tau = \frac{\hat{l}_{q,t}}{\hat{l}_q}$ , where  $\hat{l}_{q,t}$  is the estimated vehicle length at frame  $t$  and  $\hat{l}_q$  the arithmetic mean value of all measurements for that specific  $q$ -th vehicle. In these experiments, the threshold is set to  $\tau = 80\%$ .

Next, a suitable time window, where only Kalman Proposals are fed into the network, has to be defined. The suggested approach is to set the trigger according to the maximum velocity observed for all vehicles in relation to the average vehicle length  $\hat{l}$  and frame rate  $f_f$ :





$$\gamma = \frac{\max(v)}{\hat{l}} f_f. \quad (39)$$

Each video is initialized with 50 frames to compute a first valid  $\gamma$ . Then, the RPN is turned off according to  $\gamma$  and turned on for between 3 and 20 frames, as depicted in Table 3 on the left side. Next, the RPN is turned on for 10 frames and its off cycle is varied up to a 75 frames, as depicted in Table 3 on the right side. The results show for experiments with  $\gamma$  a hit rate of over 99%. Furthermore, shorter switching cycles show better performance. Expanding the Kalman



■ **Table 4** Execution time in milliseconds. With Kalman Proposals, the speed gain is achieved at the RPN and ROI stage. Lowering the resolution by half reduces the execution by approximately two thirds.

Resolution [px]	Registration		Detection with RPN					Tracking	Total Runtime
	SURF	ORB	Backbone	RPN/KP	ROI Heads	Pre+ Post	CUDA Total		
1920x1080	90	38	57	17	8	12	94	4	136
960x540	33	18	16	7	7	4	34	4	56
Detection with Kalman Proposals									
1920x1080	90	38	57	2	4	11	74	4	116
960x540	33	18	16	2	4	4	26	4	48

\* A100 [2020]  2.9  
 RTX 3090 [2020]  1.75  
 \* V100 [2017]  1.25  
 2080 Ti [2018]  1

■ **Figure 7** Comparison between two generations of consumer and server grade (\*) Nvidia GPUs with their release date in squared brackets. The numbers indicate the relative training throughput for a Mask R-CNN network according to [33].

Proposal bounding boxes to compensate prediction uncertainty yields lower performance. For example, increasing the boxes in both axis by 5% or 10% reduces the hit rate by approximately 0.4% and 0.7%. Finally, it should be noted, that the runtime and hit rate with Kalman proposals is tested with a Mask R-CNN network. Nevertheless, these proposals are applicable for replacing the common brutal force proposal generation as part of other neural network architectures as well. The execution time for the detection is not only reduced in the region proposal part, but also in the final detection and segmentation head, since only a few proposals are evaluated by the network.

### 3.9.1 Runtime and hardware requirements

This section takes a in-depth look at the runtimes. The measurements are depicted in Table 4. The code was processed on a workstation<sup>5</sup>. Image registration was performed with the ORB and SURF implementation from the OpenCV library. The measurement includes the writing of registered images on the hard drive. ORB outperforms SURF in terms of execution time. Nevertheless, the registration acquires 38 ms for a FHD resolution. The other major bottlenecks are the backbone (ResNet50-FPN) and the RPN. The average RPN execution time can be dramatically reduced by using the Kalman Proposals. The pre- and post-processing part within the detection performs image normalization, copies the image into the GPU RAM and expands the predictions to the input image size after the network again.

Regarding the object detection, the Kalman Proposals reduce the time for the proposal generation, as well as for the ROI heads. The proposal generation takes only 2 ms instead of 17 ms, and the ROI heads runtime is reduced by half. The runtime is reduced to about one third by decreasing the resolution by half. The tracking part includes copying the results to the hard disk,

<sup>5</sup> Intel Xeon E-2176G, Nvidia Geforce 2080 Ti, M.2 SSD

the code is written in Python. Overall, the complete method achieves approximately 7 fps at FHD and 18 fps at half FHD resolution, considering the workstation in use. A benchmark from [33] indicates, that the throughput for a Mask R-CNN network increases by about 75% with the next generation GPU, compared to the GPU used in this work, see Figure 7. With new generation hardware, typical camera frame rates of 25 or 30 fps are within reach for half FHD resolution.

In order to achieve a higher throughput, besides faster hardware, several aspects can be considered for future work: 1) perform the registration on the GPU directly and pass the image tensor on to the detection network, 2) avoid image registration by detecting Ground Control Points automatically in order to apply the image transformation, 3) replace the backbone by a light-weight network.

On-board GPU accelerated hardware, mounted on a drone, can only process a fraction of the information compared to a full-size GPU. Therefore, running high resolution image registration and segmentation on such platforms remains a challenging task currently.

## **4 Macroscopic statistics**

In the previous section the focus was to detail how to generate new traffic data sets by using drones. The focus of the following is to show how such data sets can be used to perform a macroscopic traffic analysis. The pre-processed image data from the publicly available highD dataset [28] is used because such type of analysis benefit from bigger data sets. The results are compared with literature data to examine the validity of the approach.

### **4.1 Data set description**

The highD data set was published alongside with the publication [28]. Videos of German motorways were recorded from a drone perspective. The following list, adopted from [28], depicts some key facts about the data set:

- around 110 500 vehicles recorded,
- road length of about 420 m with two or three lanes,
- the total driven distance is 45 000 km,
- the GSD is 0.1 m/px.

The semantic segmentation was performed using a modified U-Net neural network [48]. Transfer learning was applied with around 3000 manually labeled image patches. The trajectories were smoothed and validated in a post-processing step according to the authors. Already preprocessed variables are for example the velocities, accelerations and time headway for each recorded frame. The vehicles are divided into two classes: cars and trucks.

The present work focuses on the aggregated, macroscopic view on the data. For an example for utilizing the data for a vehicle based view, interested readers are referred to [51]. In that work, cut-ins and hard braking maneuvers are analyzed and related to automatic emergency braking system alerts.

### **4.2 Variables of interest**

The three macroscopic variables considered in this work are the traffic flow rate, the traffic density and the average velocity of the traffic stream. With these variables the fundamental diagrams are plotted. The flow rate  $q$  is defined as

$$q = \text{vehicles}/\text{time} \tag{40}$$

and is measured at a certain point over time. The traffic density  $\rho$  is defined as

$$\rho = \text{vehicles}/\text{distance}. \quad (41)$$

In order to measure the traffic density, a road segment of at least several hundred meters should be observed [20]. The third and last variable, the average velocity, can be estimated as “time mean velocity” or “space mean velocity” [20]. The difference of both is minor for stable traffic flow and therefore neglected in the following.

Two more relevant variables for traffic modeling are the distance to the front vehicle, the distance headway (DHW), and the time headway (THW). The THW is defined as

$$THW = \frac{x_l - x_f}{v_f}, \quad (42)$$

where  $x$  denotes the position and  $v$  the velocity. The subscript  $l$  denotes the leader vehicle, the subscript  $f$  the follower vehicle.

### 4.3 Fundamental diagrams

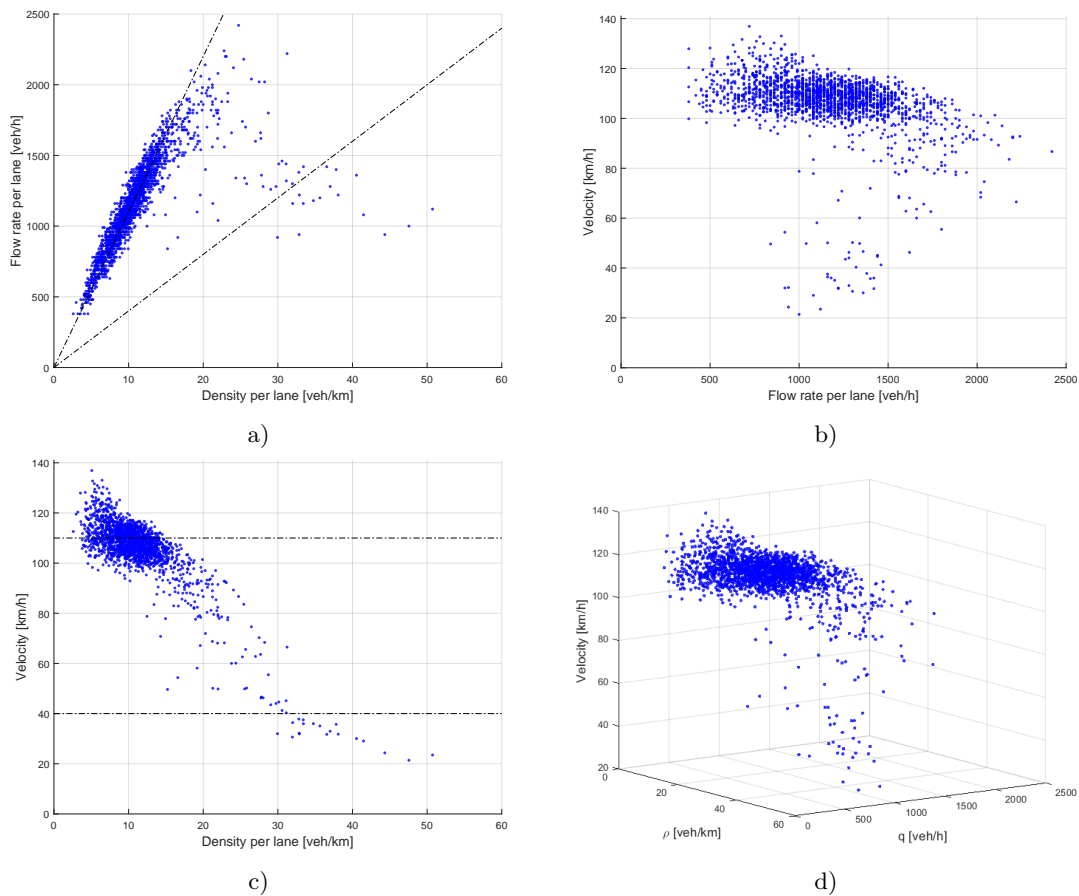
This subsection starts with a brief description of the terms free flow, bounded flow and congested traffic. Graphically, these terms are depicted in the fundamental traffic diagrams. The fundamental diagrams visualize the relationship of traffic flow, density and speed. In this work, these macroscopic variables are also linked to the microscopic based THW measure in order to demonstrate the interconnection between both domains. A synchronous linking of both domains is difficult with traditional methods, but can be easily achieved with drone based image data analysis.

The terms free flow, bounded flow and congested/jammed traffic are briefly discussed based on [43] and referenced literature. Graphically, they are depicted by the dashed lines in the  $\rho - q$  diagram in Figure 8 a).

During free flow traffic, the dependency of the flow rate to the density can be fitted linearly, as the fluctuations are minor. The positive signed slope of the curve represents the average velocity for that traffic type. Exceeding a certain density threshold, the average velocity drops significantly. The critical density threshold depends on the drivers and environmental parameters, including the road infrastructure. Beyond the critical density threshold, the traffic characteristics change and reach the state of bounded traffic. Bounded traffic can be divided into three classes. First, the homogeneous flow, where density and velocities stay rather constant. This state can be depicted as a point in the fundamental diagram. In the second case, the velocity is homogeneous, but the flow rate and density vary over time and space. This state can be depicted as a line with negative gradient in the fundamental diagram. In the third and most frequently appearing class for bounded traffic, all three parameters vary, while vehicles still keep a certain velocity. Considering time series data, the consecutive datapoints of this inhomogeneous traffic stream class jump stochastically in the fundamental diagram. A further increase of the traffic density yields a stop-and-go situation followed by the jammed traffic with zero velocity.

The following part of this subsection depicts the classical fundamental diagrams derived from the data set. The relationship between the density and flow rate is depicted in Figure 8 a). For free flow traffic up to around  $\rho = 10 - 15 \text{ veh/km}$ , the curve can be fitted linearly. The correlation coefficient  $R_c$  for  $\rho < 15 \text{ veh/km}$  is  $R_c = 0.94$ . The zone for bounded traffic up to stop-and-go traffic is depicted within the two dashed lines. The transition from stop-and-go to jammed traffic arises with a density of approximately  $\rho = 30 - 40 \text{ veh/km}$ , while saturating at around  $\rho = 45 - 50 \text{ veh/km}$  per lane. The plot coincides with the results of the empirical studies in [14, 43], which were obtained with another data source.

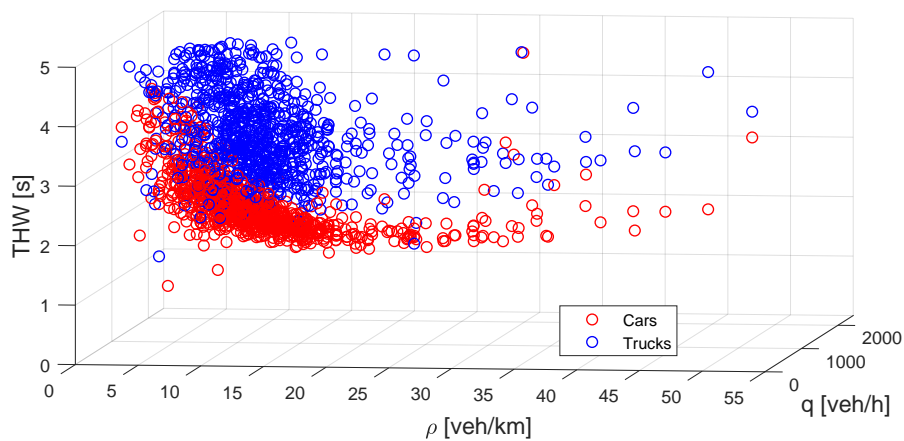
## 02:22 Drone Image Data For Joint Micro- and Macroscopic Road Traffic Analysis



■ **Figure 8** Fundamental diagrams: The dashed lines in a) and c) represent the transition from free flow to bounded flow up to stop-and-go traffic. Jammed traffic was not recorded within one-minute periods.

Figure 8 b) depicts the average flow rate over the velocity. Two different velocity ranges can be found at the same flow rate, which yields to the classification of stable and unstable traffic flow. Figure 8 c) depicts the velocity over the traffic density. The graph is approximately divided into the three traffic types (dashed lines). The average velocity decreases slightly up to a certain density threshold. Above the threshold the velocity drops significantly. Most of the recorded data is located in the transition zone of free flow to bounded traffic, while still retaining a velocity of  $v \geq 100 \text{ km/h}$ . Figure 8 d) depicts a three-dimensional plot, combining all three variables.

Finally, Figure 9 depicts the relationship between the flow rate, density and THW from all available time frames, aggregated to one minute slices. Overall, the THW is considerably lower for cars than for trucks. The datapoints for cars show a clear tendency towards smaller THW with an increasing density and flow rate. Trucks have a larger spread and are less correlated to the density and flow. The THW values reach a bottom of 1.2s around the transition zone between bounded traffic and stop-and-go traffic ( $\rho \approx 30 \text{ veh/km}$ ). From there on a tendency towards larger THW values can be recognized. Drivers keep in average a certain minimum distance of some meters to their leader vehicle during stop-and-go traffic. This yields larger averaged THW values due to the low velocity.



■ **Figure 9** The traffic density, flow rate and the average THW over all recorded time frames, aggregated to one minute slices.

#### 4.4 Lane changes and load per lane

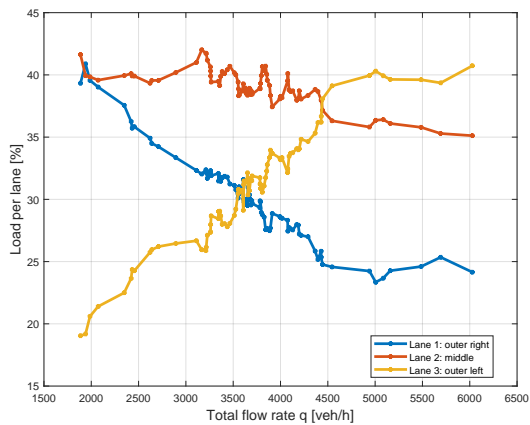
For the design of highways, it is important to know to which proportions a lane is used and where the saturation point is reached. From the perspective of traffic safety, accidents caused by lane changes are ranked third, behind speed violations and short distances [1, 2]. Since these two indicators are important for traffic analysis, the load per lane and lane changes in dependence to the traffic flow are examined and compared to literature.

Figure 10 depicts the relative lane load depending on the traffic flow rate of the road. Here, the load per lane is defined as the fraction of the overall number of vehicles on the road segment. Higher traffic flow rates increase the proportion of vehicles on the outer left lane. Smaller flow rates increase the proportion to the outer right lane, since drivers are legally obliged to drive on the right lane, if not driving at a higher velocity than the leader vehicle. In order to reach velocities above the truck velocity limits at higher flow rates, drivers switch to the middle and with further increasing traffic flow switch more often to the left lane. This observation holds as long as the traffic does not reach a stop-and-go state. The conclusions from [14] coincide by means of the tendency, that higher flow rates increase the proportion of vehicles on the left lane, decrease it on the middle lane slightly and decreases it heavily on the right lane. The publication [14] proposes flow rates per lane as depicted in Figure 11. The right lane saturates at around  $q = 800$  veh/h and the left lane at around  $q = 2600$  veh/h [14, 29].

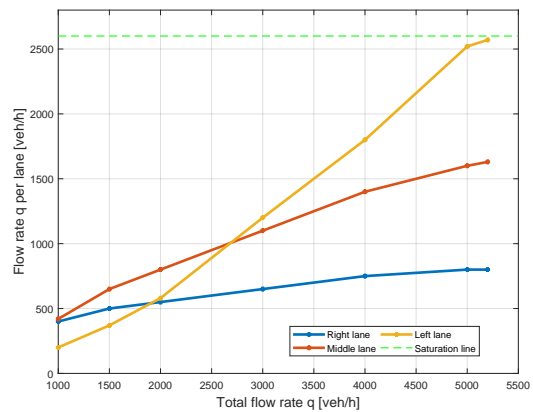
Figure 12 and Figure 13 depict the normalized number of lane changes over the flow rate and traffic density. The data points depicted in the figures are aggregated values of one-minute slices.

Publication [14] proposes a maximum lane change rate at around 3500 vehicles per hour for a 3 lane motorway. In [14] the lane change rate is described by step functions which form a triangular shape. Similar to [14], a triangular fit for the highD data set is depicted in Figure 12. It encloses at least 97% of all depicted one-minute slices for the upper and lower road. The results on the lane change rate coincide with the observation in [14]. Regarding the dependency on the traffic density, the lane change rate increases up to around 10-12 vehicles per kilometer and lane [14], according to the findings depicted in Figure 13.

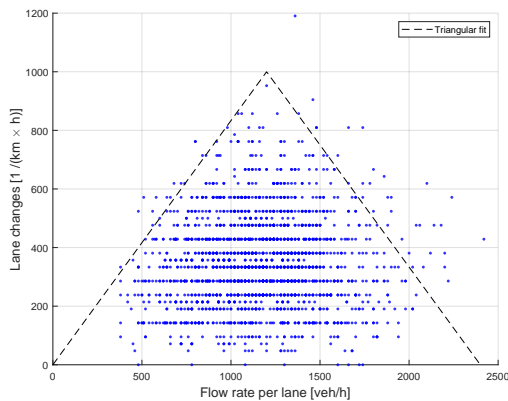
The results from this chapter confirm that the drone-based approach can provide valid estimates on macroscopic traffic data. Furthermore, the results can be interpreted in the context of microscopic variables as well. This is demonstrated with the relationship of the THW to the traffic density and flow.



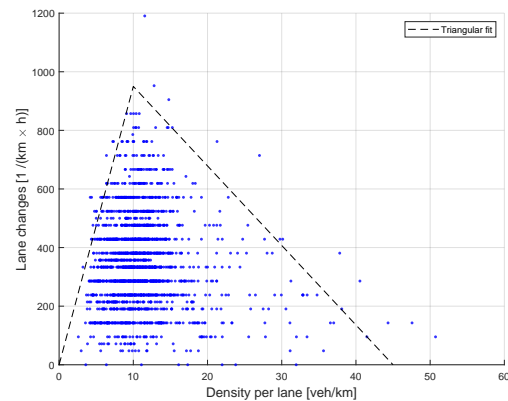
■ **Figure 10** The relative lane load depending on the averaged traffic flow rate. The curves are processed with a smoothing filter. For this plot only roads with 3 lanes are considered.



■ **Figure 11** A schematic comparison of the average load for a 3 lane motorway from [14]. The right lane saturates around  $q = 800 \text{ veh/h}$ , the left lane around  $q = 2600 \text{ veh/h}$ .



■ **Figure 12** The number of lane changes per lane, hour and kilometer depending on the averaged traffic flow rate per lane.



■ **Figure 13** The number of lane changes per lane, hour and kilometer depending on the traffic density.

## 5 Conclusions

The technological improvement of drones combined with the capabilities of computer vision methods, namely deep neural networks, open a new chapter for applications in road traffic surveillance and analysis. Observation campaigns can be carried out at every place, without the need of installing and maintaining roadside units. First projects attempt to expand the field of view by operating with swarms of drones and thereby capture whole city districts. The accuracy of the state estimation by means of aerial imagery is suitable for many trajectory based applications as well. This enables the individual analysis of objects, as well as observing the interaction between traffic participants.

In this work, a methodology for generating microscopic traffic data is proposed. More precisely, the state of vehicles and the resulting trajectories are estimated. The method is validated with experiments and reaches consumer-grade INS precision for the vehicle state estimation. A detailed look into the error sources, limitations and runtimes complements the proposed method.

Additionally, a new approach for reducing the average computational runtime, named Kalman Proposals, are presented. With Kalman Proposals, the runtime for object detection can be decreased by up to 20% with minor losses on the detection performance.

Furthermore, a publicly available data set for highway traffic is analyzed in order to validate the drone based approach for macroscopic traffic analysis. The statistics derived in this work are compared to related publications, where data was collected and processed with alternative approaches. The analysis confirms the applicability for capturing macroscopic traffic data as well.

In conclusion, the results of this paper underline the versatility of drone based image processing for synchronous macro- and microscopic road traffic analysis.

## References

- 1 Daten und Fakten: Autobahn-Unfälle, 2010.
- 2 Runter vom Gas, Unfallursachen, 2018.
- 3 Mujahid Abdulrahim. On the dynamics of automobile drifting. *SAE Mobilus*, April 2006. doi:10.4271/2006-01-1019.
- 4 Seyed Majid Azimi, Eleonora Vig, Reza Bahmanyar, Marco Körner, and Peter Reinartz. Towards multi-class object detection in unconstrained remote sensing imagery, 2018. arXiv:1807.02700.
- 5 S. Bae and K. Yoon. Robust online multi-object tracking based on tracklet confidence and online discriminative appearance learning. In *2014 IEEE Conference on Computer Vision and Pattern Recognition*, pages 1218–1225, 2014. doi:10.1109/CVPR.2014.159.
- 6 Emmanouil Barmponakis and Nikolas Geroliminis. On the new era of urban traffic monitoring with massive drone data: The pneuma large-scale field experiment. *Transportation Research Part C: Emerging Technologies*, 111:50–71, 2020. doi:10.1016/j.trc.2019.11.023.
- 7 Herbert Bay, Tinne Tuytelaars, and Luc Van Gool. SURF: Speeded Up Robust Features. In *Computer Vision – ECCV*, 2019.
- 8 A. Bewley, Z. Ge, L. Ott, F. Ramos, and B. Uppcroft. Simple online and realtime tracking. In *2016 IEEE International Conference on Image Processing (ICIP)*, 2016.
- 9 Peter J. Bickel, Chao Chen, Jaimyoung Kwon, John Rice, Erik van Zwet, and Pravin Varaiya. Measuring Traffic. *Statistical Science*, 22(4), 2007.
- 10 Laura Bieker-Walz. Wie kann eine Verkehrssimulation den Rettungsdienst unterstützen? In *WAW DLR.Open II*, oktober 2017. URL: <https://elib.dlr.de/114841/>.
- 11 Ilker Bozcan and Erdal Kaya. Au-air: A multimodal unmanned aerial vehicle dataset for low altitude traffic surveillance. In *IEEE International Conference on Robotics and Automation (ICRA)*, 2020.
- 12 Antonia Breuer, Jan-Aike Termöhlen, Silviu Homoceanu, and Tim Fingscheidt. Opended: A large-scale roundabout drone dataset. In *Proceedings of International Conference on Intelligent Transportation Systems*, September 2020.
- 13 Bundesverband der Deutschen Luftverkehrswirtschaft. Analyse des deutschen Drohnenmarktes. URL: <https://www.bdl.aero/de/publikation/analyse-des-deutschen-drohnenmarktes/>.
- 14 Fritz Busch. Spurbelastungen und Häufigkeit von Spurwechseln auf einer dreispurigen BAB-Richtungsfahrbahn. *ATZ - Automobiltechnische Zeitschrift*, June 1984. doi:10.1007/s35148-012-0485-x.
- 15 CATT (University of Maryland). Traffic Flow Measures Implementation Guide, 2008. URL: [http://www.catt.umd.edu/sites/default/files/documents/traffic\\_flow\\_measure\\_guidelines\\_v8.pdf](http://www.catt.umd.edu/sites/default/files/documents/traffic_flow_measure_guidelines_v8.pdf).
- 16 Bo-Chiuan Chen and Feng-Chi Hsieh. Sideslip angle estimation using extended kalman filter. *Vehicle System Dynamics - VEH SYST DYN*, 46, September 2008. doi:10.1080/00423110801958550.
- 17 C. Feichtenhofer, A. Pinz, and A. Zisserman. Detect to track and track to detect. In *2017 IEEE International Conference on Computer Vision (ICCV)*, pages 3057–3065, 2017. doi:10.1109/ICCV.2017.330.
- 18 B Grienshields. The photographic method of studying traffic behavior. In *Proceedings of the Thirteenth Annual Meeting of the Highway Research Board*, 1933.
- 19 Giuseppe Guido, Vincenzo Gallelli, Daniele Rogano, and Alessandro Vitale. Evaluating the accuracy of vehicle tracking data obtained from Unmanned Aerial Vehicles. *International Journal of Transportation Science and Technology*, 2016.
- 20 Hall, Fred. TRAFFIC STREAM CHARACTERISTICS, 1996.
- 21 Jörg Haus and Norbert Lauinger. Optische gitter: Die abbildung der realität – 75 jahre berührungslose dynamische meßtechnik auf der basis optischer gitter. *Laser Technik Journal*, 4:43–47, April 2007. doi:10.1002/latj.200790155.
- 22 Kaiming He, Georgia Gkioxari, Piotr Dollár, and Ross Girshick. Mask R-CNN. In *Proceedings of the International Conference on Computer Vision (ICCV)*, 2017.
- 23 Dirk Helbing. Traffic and related self-driven many-particle systems. *Rev. Mod. Phys.*, 73:1067–1141, December 2001. doi:10.1103/RevModPhys.73.1067.
- 24 R.E. Kalman. A new approach to linear filtering and prediction problems, 1960.
- 25 Kai Kang, Hongsheng Li, Tong Xiao, Wanli Ouyang, Junjie Yan, Xihui Liu, and Xiaogang Wang. Object detection in videos with tubelet proposal networks. *2017 IEEE Conference on Com-*



- puter Vision and Pattern Recognition (CVPR), July 2017. doi:10.1109/cvpr.2017.101.
- 26 Kai Kang, Wanli Ouyang, Hongsheng Li, and Xiaogang Wang. Object detection from video tubellets with convolutional neural networks. *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2016. doi:10.1109/cvpr.2016.95.
  - 27 Ebrahim Karami, Siva Prasad, and Mohamed Shehata. Image Matching Using SIFT, SURF, BRIEF and ORB: Performance Comparison for Distorted Images, 2017.
  - 28 Robert Krajewski, Julian Bock, Laurent Kloeker, and Lutz Eckstein. The highD Dataset: A Drone Dataset of Naturalistic Vehicle Trajectories on German Highways for Validation of Highly Automated Driving Systems. In *IEEE 21st International Conference on Intelligent Transportation Systems (ITSC)*, 2018.
  - 29 Stefan Krauß. Microscopic Modeling of Traffic Flow: Investigation of Collision Free Vehicle Dynamics.
  - 30 F. Kruber, E. Sánchez Morales, S. Chakraborty, and M. Botsch. Vehicle Position Estimation with Aerial Imagery from Unmanned Aerial Vehicles. In *IEEE Intelligent Vehicles Symposium (IV)*, 2020.
  - 31 H. W. Kuhn and Bryn Yaw. The hungarian method for the assignment problem. *Naval Res. Logist. Quart.*, pages 83–97, 1955.
  - 32 C. Kyrkou, G. Plastiras, T. Theocharides, S. I. Venieris, and C. Bouganis. DroNet: Efficient convolutional neural network detector for real-time UAV applications. In *Design, Automation Test in Europe Conference Exhibition (DATE)*, 2018.
  - 33 Lambda Labs. Deep Learning GPU Benchmarks, 2021. URL: <https://lambdalabs.com/gpu-benchmarks>.
  - 34 Qingpeng Li, Lichao Mou, Qizhi Xu, Yun Zhang, and Xiao Xiang Zhu. R<sup>3</sup>-Net: A Deep Network for Multi-oriented Vehicle Detection in Aerial Images and Videos. *CoRR*, 2018. arXiv:1808.05560.
  - 35 H. Lietz, Petzoldt T., M. Henning, J. Haupt, G. Wanielik, J. Krems, H. Mosebach, J. Schomerus, M. Baumann, and U. Noyer. Methodische und technische Aspekte einer Naturalistic Driving Study. *FAT Schriftenreihe*, 2011.
  - 36 Thomas Lillesand, Ralph W. Kiefer, and Jonathan Chipman. *Remote Sensing and Image Interpretation*, volume 5. Wiley, 2003.
  - 37 T. Lin, P. Dollár, R. Girshick, K. He, B. Hariharan, and S. Belongie. Feature Pyramid Networks for Object Detection. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017.
  - 38 Tsung-Yi Lin, Michael Maire, Serge Belongie, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollár, and C. Lawrence Zitnick. Microsoft COCO: Common Objects in Context. In *Computer Vision – ECCV*, 2014.
  - 39 J. Long, E. Shelhamer, and T. Darrell. Fully convolutional networks for semantic segmentation. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2015.
  - 40 A.L. Majdik, C. Till, and D. Scaramuzza. The zurich urban micro aerial vehicle dataset. *International Journal of Robotics Research*, 2017.
  - 41 L. Mou and X. X. Zhu. Vehicle Instance Segmentation From Aerial Image and Video Using a Multitask Learning Residual Fully Convolutional Network. *IEEE Transactions on Geoscience and Remote Sensing*, 2018.
  - 42 Matthias Mueller, Neil Smith, and Bernard Ghanem. A benchmark and simulator for uav tracking. In *Computer Vision – ECCV 2016*. Springer International Publishing, 2016.
  - 43 Lutz Neubert. *Statistische Analyse von Verkehrsdaten und die Modellierung von Verkehrsfluss mittels zellulärer Automaten*. PhD thesis, Universität Duisburg, 2000.
  - 44 Pix4D SA. Example projects - real photogrammetry data (<https://www.pix4d.com>). URL: <https://www.pix4d.com>.
  - 45 J. Redmon, S. Divvala, R. Girshick, and A. Farhadi. You Only Look Once: Unified, Real-Time Object Detection. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016.
  - 46 Shaoqing Ren, Kaiming He, Ross Girshick, and Jian Sun. Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks. In *Advances in Neural Information Processing Systems 28*. Curran Associates, Inc., 2015.
  - 47 A. Robicquet, A. Sadeghian, A. Alahi, and S. Savarese. Human Trajectory Prediction In Crowded Scenes. In *European Conference on Computer Vision (ECCV)*, 2016.
  - 48 Olaf Ronneberger, Philipp Fischer, and Thomas Brox. U-net: Convolutional networks for biomedical image segmentation. *CoRR*, 2015.
  - 49 E. Rublee, V. Rabaud, K. Konolige, and G. Bradski. Orb: An efficient alternative to sift or surf. In *2011 International Conference on Computer Vision*, 2011.
  - 50 E. Sánchez Morales, F. Kruber, M. Botsch, B. Huber, and A. García Higuera. Accuracy Characterization of the Vehicle State Estimation from Aerial Imagery. In *IEEE Intelligent Vehicles Symposium (IV)*, 2020.
  - 51 Patrick Schneider, Martin Butz, Christian Heinzemann, Jens Oehlerking, and Matthias Woehrle. Scenario-based threat metric evaluation based on the highd dataset. In *IEEE Intelligent Vehicles Symposium (IV)*, 2020.
  - 52 Dieter Schramm, Manfred Hiller, Roberto Bardini, et al. *Modellbildung und Simulation der Dynamik von Kraftfahrzeugen*, volume 124. Springer, 2010.
  - 53 Marc René Zofka Tobias Fleck, Sven Ochs and J. Marius Zöllner. (accepted) robust tracking of reference trajectories for autonomous driving in intelligent roadside infrastructure. In *Intelligent Vehicles Symposium (IV) 2020*, oktober 2020.
  - 54 Philip Torr and A. Zisserman. MLESAC: A New Robust Estimator with Application to Estimating Image Geometry. *Computer Vision and Image Understanding*, June 2000.
  - 55 Verband der TÜV e.V. Merkblatt 751, 2008.
  - 56 Bas Vergouw, Huub Nagel, Geert Bondt, and Bart Custers. *Drone Technology: Types, Payloads, Applications, Frequency Spectrum Issues and Future Developments*, pages 21–45. T.M.C. Asser Press, 2016. doi:10.1007/978-94-6265-132-6\_2.



- 57 L. Wang, W. Ouyang, X. Wang, and H. Lu. Visual tracking with fully convolutional networks. In *2015 IEEE International Conference on Computer Vision (ICCV)*, pages 3119–3127, 2015. doi: 10.1109/ICCV.2015.357.
- 58 Wei Zhan, Liting Sun, Di Wang, Haojie Shi, Aubrey Clause, Maximilian Naumann, Julius Kümmerle, Hendrik Königshof, Christoph Stiller, Arnaud de La Fortelle, and Masayoshi Tomizuka. INTERACTION Dataset: An INTERnational, Adversarial and Cooperative moTION Dataset in Interactive Driving Scenarios with Semantic Maps. *arXiv*, 2019. arXiv:1910.03088.
- 59 Pengfei Zhu, Longyin Wen, Xiao Bian, Ling Haibin, and Qinghua Hu. Vision Meets Drones: A Challenge. *arXiv preprint*, 2018. arXiv:1804.07437.

# HW-Flow: A Multi-Abstraction Level HW-CNN Codesign Pruning Methodology

Manoj-Rohit Vemparala ✉ 

BMW Autonomous Driving, Munich, Germany

Alexander Frickenstein ✉

BMW Autonomous Driving, Munich, Germany

Manfredi Camalleri ✉ 

BMW Autonomous Driving, Munich, Germany

Christian Unger ✉

BMW Autonomous Driving, Munich, Germany

Maurizio Martina ✉ 

Politecnico di Torino, Turin, Italy

Nael Fafous ✉ 

Technical University of Munich, Munich, Germany

Emanuele Valpreda ✉ 

Politecnico di Torino, Turin, Italy

Qi Zhao ✉ 

BMW Autonomous Driving, Munich, Germany

Naveen-Shankar Nagaraja ✉ 

BMW Autonomous Driving, Munich, Germany

Walter Stechele ✉ 

Technical University of Munich, Munich, Germany

## Abstract

Convolutional neural networks (CNNs) have produced unprecedented accuracy for many computer vision problems in the recent past. In power and compute-constrained embedded platforms, deploying modern CNNs can present many challenges. Most CNN architectures do not run in real-time due to the high number of computational operations involved during the inference phase. This emphasizes the role of CNN optimization techniques in early design space exploration. To estimate their efficacy in satisfying the target constraints, existing techniques are either hardware (HW) agnostic, pseudo-HW-aware by considering parameter and operation counts, or HW-aware through inflexible hardware-in-the-loop (HIL) setups. In this work,

we introduce HW-Flow, a framework for optimizing and exploring CNN models based on three levels of hardware abstraction: *Coarse*, *Mid* and *Fine*. Through these levels, CNN design and optimization can be iteratively refined towards efficient execution on the target hardware platform. We present HW-Flow in the context of CNN pruning by augmenting a reinforcement learning agent with key metrics to understand the influence of its pruning actions on the inference hardware. With  $2\times$  reduction in energy and latency, we prune ResNet56, ResNet50, and DeepLabv3 with minimal accuracy degradation on the CIFAR-10, ImageNet, and CityScapes datasets, respectively.

**2012 ACM Subject Classification** Computing Methodologies → Artificial intelligence

**Keywords and Phrases** Convolutional Neural Networks, Optimization, Hardware Modeling, Pruning

**Digital Object Identifier** 10.4230/LITES.8.1.3

**Received** 2020-12-15 **Accepted** 2021-09-05 **Published** 2022-11-16

**Editor** Samarjit Chakraborty and Qing Rao

**Special Issue** Special Issue on Embedded Systems for Computer Vision

## 1 Introduction

Convolutional neural networks (CNN) are widely used for solving problems like image classification [13], semantic segmentation [3], object detection [45], complex autonomous driving tasks [2] and medical diagnosis of brain tumors [28]. Having outperformed hard-coded algorithms on challenging benchmarks such as ImageNet [30], CNNs also surpassed human-level accuracy. However, the computational complexity of these networks hampers their application in embedded environments. Most accurate CNN models require up to hundreds of megabytes for parameter storage [24] and billions of multiplications [32]. For instance, a ResNet-152 [13] trained on the ImageNet dataset [30] requires up to 244MB of learned parameters to execute 517 layers, with around 22 billions operations. Compression techniques have become an essential topic of research

for finding light-weight architectures capable of efficiently solving various deep learning tasks. Despite the remarkable compression rates of existing pruning methods, conventional approaches are either hardware (HW) agnostic, pseudo-HW-aware by considering proxies, or HW-aware through inflexible hardware-in-the-loop (HIL) setups, which lead to vendor lock-ins.

In embedded applications such as autonomous driving and robotics, the design of neural networks and the target HW accelerator goes hand in hand. During the early development phases, it is likely that the target platform is not fully defined, the HW is not available, or compilers are prone to errors, making a HIL-based approach challenging. Alternatively, proxy metrics, such as parameter (**Param**) and operation (**OP**) counts, offer a detached yet loosely correlated indication of hardware performance [1]. Reliance on proxy metrics oversimplifies the problem at hand and does not always guarantee improvements in energy or latency when deployed on real hardware. Directly choosing a hardware-platform restricts the CNN optimizer, and conversely, directly choosing the compression technique may not match the hardware architectures being designed.

This circumstance calls for a HW-CNN codesign paradigm that guarantees synergies in the process of deploying CNNs to real-world applications. This work aims to estimate the implementation metrics at different abstraction levels through various hardware models and a novel scheduler. This allows for either a top-down or a meet-in-the-middle codesign approach, giving the designer the ability to gradually traverse through the design abstraction levels, while permitting design space exploration and exploitation after each stage of refinement. With this flexibility, the designer can analyze the impact of various CNN pruning configurations and HW specific hyperparameters, without committing to a target hardware platform in the early design phases. This ultimately leads to a platform-aware optimization technique, which improves energy efficiency and/or latency at design time.

We remove the limitations of pure proxy and HIL-based neural network pruning and use a reinforcement learning (RL) agent to prune filters by considering the estimates of the proposed HW-Flow framework. With this method, we overcome the burden of wasting GPU-hours for optimizing CNNs, which do not guarantee an efficiency gain for a target hardware platform. We can assert that the decision of pruning rate for each layer is highly correlated to the target hardware constraint and inference platform. The estimates generated for a HW platform can directly influence *which* layers are pruned and to what extent. They also help the CNN designer understand which scheduling schemes and hardware dimensions are necessary to have a reasonable pruning rate/burden, to match the target application constraints. The contributions of this work can be summarized as follows:

- We introduce HW-Flow, a framework for optimizing and exploring CNN models based on three hardware abstraction levels, *Coarse* | *Mid* | *Fine*, by scheduling and mapping workloads onto potential HW-architectures. With this approach, we model different HW platforms without costly fabrication and explore the pruning potential of CNN models at each design phase.
- We reduce the time required to produce an optimal schedule using analytical search approaches, circumventing exhaustive and random sampling techniques used in literature.
- We augment a state-of-the-art learning-based pruning agent [16] with rewards in the form of model-based HW estimates (e.g., **OPs**, DRAM accesses, energy, and latency). Using this information, the agent produces an optimal pruning strategy required to meet the target constraints. We obtain different pruning configurations, which result in a  $2\times$  reduction of the respective KPIs (Key Performance Indicators), with minimal accuracy degradation.

## 2 Background

### 2.1 Convolutional Neural Networks

CNNs are deep neural networks which are well-suited for generating predictions based on multi-dimensional, localized input feature spaces, e.g. image processing applications. The convolution of an input activation  $A^{l-1}$  with the convolution kernel  $W^l$  produces an output feature map  $A^l$ , where each pixel of the feature map  $A^l$  can be computed as shown in equation 1.

$$A^l[c_o][h_o][w_o] = \sum_{c_i}^{\overbrace{C_i}^{\text{Inp.Ch}}} \sum_{k_w}^{\overbrace{K_w}^{\text{Kernel.dim}}} \sum_{k_h}^{\overbrace{K_h}^{\text{Kernel.dim}}} a_{c_i, w_o \cdot s + k_w, h_o \cdot s + k_h}^{l-1} \cdot w_{c_o, c_i, k_w, k_h}^l, \text{ where } A^l \in \mathbb{R}^{C_o \times H_o \times W_o} \quad (1)$$

The input feature maps (*Ifmaps*) denoted by  $A^{l-1}$  are composed of multiple channels  $C_i$  and spatial dimensions  $W_i, H_i$ . To compute the convolution operation, the kernel of dimensions  $K_w \times K_h$  slides across the input 2-D map with stride size  $s$ . A dot-product is performed between the kernel pixels  $w^l \in W^l$  and a sub-set of pixels  $a^{l-1} \in A^{l-1}$  from the input volume. The dot-product accumulates the values across all input channels resulting in an output pixel. The convolution operation is the repetition of the aforementioned dot-product operation for the entire *Ifmap* with  $C_o$  filters, generating output feature maps (*Ofmaps*)  $A^l \in \mathbb{R}^{W_o \times H_o \times C_o}$ . Successive layers detect different features in the input image at different scales. The first layers are usually responsible of recognizing simple shapes, edges and patterns, while complex features can be detected at the deeper stages of the network. Fully Connected (FC) layers can be simplified considered a special case of the convolution operation by setting  $W_i = K_w, H_i = K_h, W_o = 1$  and  $H_o = 1$ . These layers restrict weight reuse opportunities and demand high memory bandwidth during the inference. CNNs have produced better predictions than humans on computer vision applications such as image classification [13] and semantic segmentation [3] using supervised ground truth labels.

**Image Classification.** Out of  $O$  possible classes, the input image is predicted based on the output  $\tilde{Y} \in \mathbb{R}^O$ . It is typical to translate the problem into predicting the probability of each possible class given an input image, so that the output layer produces a vector with a fixed dimension of  $O$ . Several CNN topologies were proposed in the last decade to solve the image classification problem, dealing with different datasets such as CIFAR-10 [19] and ImageNet [30]. For instance, AlexNet was introduced by Krizhevsky et al. [20] as the first CNN topology for classifying the ImageNet dataset. The network consists of five convolutional layers, max-pooling layers, dropout layers and three fully-connected layers, where the last one maps to a 1000-element vector representing the number of possible classes for the ImageNet dataset. Other examples which followed in the next years include VGG-16 [24], Inception-Net [33], ResNet [13] and EfficientNet [34].

**Image Semantic Segmentation.** Segmentation-based CNNs such as FCN [25] and DeepLab [3] predict the class of each pixel in the input image from  $O$  possible categories. The semantic maps are derived from the logits  $\tilde{Y} \in \mathbb{R}^{W \times H \times O}$  with  $O$  probability values per pixel. The CNN topology for this task follows an encoder-decoder architecture. The encoder network is a feature extractor having a similar architecture as image classification CNNs and the decoder network is a set of upsampling layers which restore the original image resolution in order to predict the pixel-wise class output. FCN uses transpose convolution to upsample features whereas DeepLab uses the bilinear upsampling method.

## 2.2 Reinforcement Learning

Reinforcement learning (RL) is an exploration-exploitation approach to optimize decisions based on interactions with a dynamic environment [18]. The optimization problem is solved using an agent which is connected to the considered environment via perception and action. The environment is characterized by its state  $\mathcal{S}$  which describes the relevant features for the decision making process. At each step, the agent generates an action  $\mathcal{A}$  which changes the state of the environment and outputs a reinforcement signal or reward  $\mathcal{R}$  describing the quality of this state transition. Given a state and action space, the purpose is to find a policy which maps each state to the optimal action, which maximizes the sum of future rewards. RL-based systems often balance a trade-off between exploration and exploitation while predicting new actions. Exploitation is entirely based on the knowledge acquired by the agent and predicts actions that maximize the expected return value according to the gained experience. Exploration assumes that the current knowledge can be further improved by exploring different actions.

Supervised learning and reinforcement learning differ from each other in two major aspects. First, supervised learning is based on a set of data pairs where each input has a clearly defined ground-truth label. This is not the case for RL where each input (action) generates an immediate reward while the objective function is to maximize the sum of all future rewards. Second, RL is typically deployed in an online manner, i.e. RL-based systems are evaluated and trained concurrently to optimize the agent decisions for new input sequences. Deep Deterministic Policy Gradient (DDPG) [23] is an RL technique that outputs continuous action using Q-Networks and Actor-Critic based policy gradients. Here Q refers to the function which the algorithm predicts. Reinforcement learning has been applied in several fields such as robotics, gaming, traffic light control, and resource management systems [26, 39, 29]. In this work, we use a DDPG-based RL agent in order to decide the optimal pruning configuration of CNNs.

## 3 Related Work

Based on the target optimization metrics, we classify pruning techniques into four categories: HW-agnostic, pseudo-HW-aware, HIL-based, and HW-modeling-based pruning techniques, as compared in Table 1. Additionally, we discuss HW-modeling works that compute the HW estimates of CNN accelerators in literature.

**HW-agnostic Pruning.** The advantages of pruning were investigated in early works such as [6, 12]. Subsequent works determined the redundant weights based on an iterative method, without considering any target hardware resource constraints, e.g. simple magnitude-based pruning [11]. Recently, He et al. [15] pruned redundant filters using a geometric median heuristic. However, the efficiency term was limited to the pruning rate (PR), i.e., the ratio of pruned to total parameters. The PR was set constant to all the layers, which does not capture the energy or latency requirements of the target inference hardware. The work by Guo et al. [10], dynamically pruned CNNs irregularly based on a saliency function during training to produce efficient networks. Recently, Frickenstein et al. [9] proposed the auto-encoder-based low-rank filter-sharing technique (ALF), which utilizes sparse auto-encoders to extract the most salient features of convolutional layers, pruning redundant filters. The above works only target to compress the CNN model with minimal accuracy degradation without considering the benefits on the target HW platform.

**Pseudo HW-aware Pruning.** The authors of [14] proposed structured channel pruning, where the saliency of individual channels is determined through Lasso regression. The pruning ratio for each layer is based on handcrafted heuristics which targets lower *proxy* metrics such as OPs and

**Params.** In more recent works, automated pruning methods have gained popularity. Huang et al. [17] trained layer-specific agents, which receive the kernel matrix as a state and produce actions to prune exact filters. Contrary to [16], here the agent has a more complex task of learning the features of a layer rather than simply its sparsity ratio. The agent’s reward is formulated using a multi-objective cost function, which aims to find CNN models with both high accuracy and low *proxy* metrics. Reward functions based on proxy metrics do not guarantee an improvement for HW deployment, as we demonstrate in the following sections. The work in [36] identifies redundant weights for different regularizations during the training process using a HW loss formulation. The HW loss is limited to optimization of *proxy* metrics.

**HW-aware Pruning.** As hardware platforms tend to be complex, the effects of arbitration, stalls, etc., may be severely understated if hardware estimations purely rely on proxy metrics. By considering real hardware metrics, hardware-in-the-loop (HIL) training frameworks have been used to verify the advantages of CNN optimization techniques pragmatically [42, 7, 16]. NetAdapt [42] prunes filters based on a preexisting look-up table of hardware metrics obtained ahead of time from a mobile device. This is a costly pseudo-HIL approach, as building the look-up table is tedious and time consuming, requiring the designer to execute all possible workloads and layer dimensions to be accurate and complete. For this method to work, the hardware would need to be decided and readily available before the CNN optimization process starts. Another drawback to the approach is that the pruning technique is performed in a layer-wise manner, which is susceptible to local minima, as inter-layer effects on the hardware platform and prediction accuracy are not considered. ChamNet [7] also adopts a look-up table strategy to estimate the latency with a Bayesian energy predictor and performs neural architectural search. The predictors for the HW metrics also require the “ready-to-use” target HW platform to perform optimization. Furthermore, if the target hardware is changed, the effort to recollect the data for the new look-up table and the Bayesian optimizer needs to be taken into account. HW-NAS-Bench [22] presents a dataset to evaluate various CNN configurations on different HW platforms. The dataset is generated by performing extensive real HW measurements on NAS-specific search spaces [8, 40] Furthermore, the dataset does not cover exploration of HW specific hyper-parameters which impacts the CNN compilation/scheduling procedures. The work in AMC-AutoML [16] demonstrated an RL pruning agent, producing channel sparsity ratios for each layer as its action after every episode. Based on the magnitude obtained from the L2-norm heuristic and the sparsity ratio of each layer given by the RL agent, the redundant channels are pruned. The work demonstrated results of both proxy metrics (OPs and Params) and HIL-based timing evaluation using TF-Lite. Another HIL-based optimization technique, HAQ [38], resorts to RL-based exploration to determine suitable, layer-wise quantization levels for weights and activations in the CNN model. The reward function, including real hardware metrics, is generated by directly executing the inference of a CNN model on a Field Programmable Gate Array (FPGA) design which supports quantized computations [31].

**Hardware Modeling.** The deterministic nature of CNN inference execution on hardware makes analytical hardware modeling an intuitive approach to simulate aspects of the synthesis and deployment phases. Timeloop [27] is a HW-modeling tool that exploits CNN execution determinism to offer accurate estimates of a given hardware description. It shows the strength of HW modeling, circumventing the need for cycle-accurate CNN hardware simulators and/or synthesized hardware in the early phases of development. The tool provides the flexibility of changing the cost of hardware operations (e.g. read, write, multiply-accumulate) and the memory hierarchy, among other design parameters. Based on the data movement constraints set by the designer, the tool searches the scheduling solution space in an exhaustive or randomly sampled manner, thereby

providing the HW estimates. The schedule search time could either last significantly long with exhaustive search or lead to a sub-optimal solution with random sampling. Interstellar [43] proposes formal dataflow definitions. Unlike Timeloop, the authors of Interstellar use the Halide programming language to represent the HW-architecture and data movement constraints. The influence of memory hierarchy and dataflows on energy efficiency and latency is investigated thoroughly. MAGNet [37] considers various CNN architectures and hardware constraints generating an optimal RTL and mapping strategy to execute the CNNs efficiently. It explores various tiling strategies and dataflows by proposing a highly configurable processing element array. Yang et al. [41] leverage a HW-model to estimate the energy requirements of each layer. The layers with the highest energy contribution present a good starting point for the pruning process, based on the L2-norm heuristic. However, energy estimates do not influence the sparsity ratio directly. The work is also limited to optimizing normalized energy, but not latency, which is an equally important parameter for real-time applications. In this work, we remove the limitations of pure proxy and HIL-based neural network pruning by introducing a multi-abstraction level HW-model for estimating the efficiency of CNN architectures. Instead of exhaustive and random sampling search techniques, we analytically reduce the size of the search space, and thereby the search-time, without sacrificing schedule efficiency. A deep deterministic policy gradient (DDPG) based learning agent is augmented with key rewards and state information, allowing it to understand the influence of its pruning actions on the inference hardware for energy and latency, and enabling HW-CNN codesign-based optimization.

■ **Table 1** Classification of pruning, modeling techniques and their advantages.

Advantage	Agnostic [15, 10, 9]	Proxy [14, 17, 36]	HIL [16, 42, 7, 22]	HW-Model [27, 41]	HW-Flow [Ours]
Accuracy optimization:	✓	✓	✓	✓	✓
OPs/Params optimization:	✗	✓	✓	✓	✓
Energy/latency optimization:	✗	✗	✓	✓	✓
HW-design exploration:	✗	✗	✗	✓	✓
Learning based agent:	✗	✓	✓	✗	✓
HW-CNN codesign (abstraction/refinement):	✗	✗	✗	✗	✓

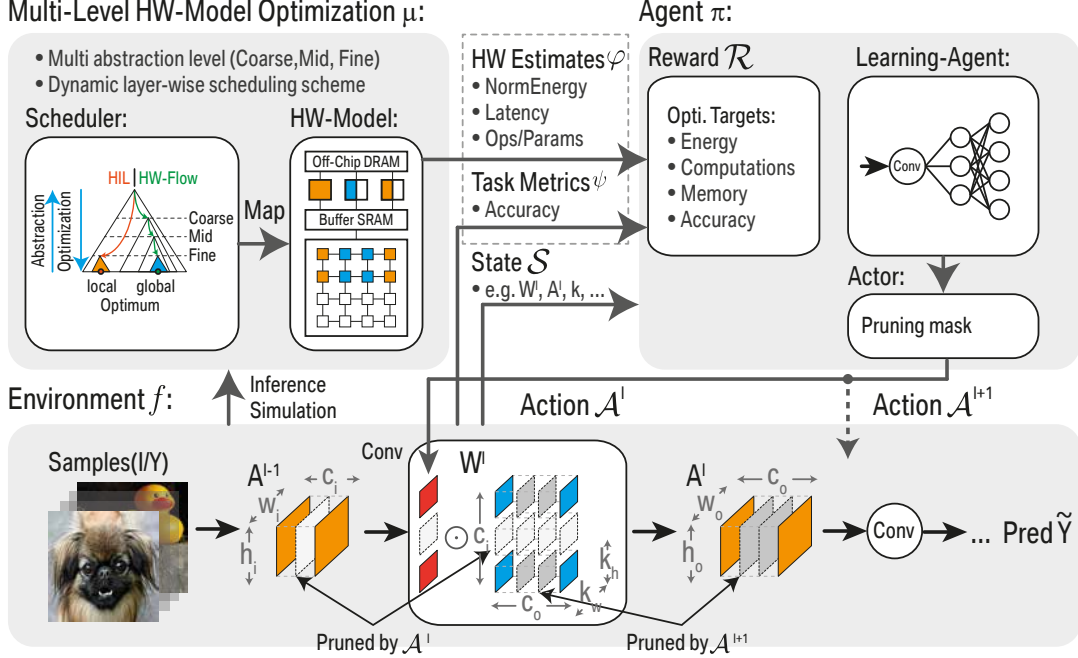
## 4 Hardware-Flow

In the case of general-purpose HW, such as off-the-shelf GPUs and CPUs, HIL-based approaches may indeed be less cumbersome than building a hardware model. For many real-time, energy, and latency-critical applications, these platforms are not applicable at deployment time. Carefully designing custom hardware, which meets specific criteria for an application, necessitates following a HW-SW codesign paradigm. In the top-down approach, this implies iteratively going through different levels of abstraction and performing some iterations of exploration before fixing some parameters and refining the design to one abstraction level lower. During inference, CNN forward-pass executions are entirely deterministic, making it hard to justify the need for synthesized hardware to observe training or optimization effectiveness. This makes hardware modeling an attractive and economical alternative for rapid prototyping and testing of different HW-aware optimization strategies.

In this paper, the HW-Flow framework is based on an interaction between the CNN environment  $f$ , the pruning agent  $\pi$  and the hardware model  $\mu$  (shown in Figure 1). In detail, the agent receives a layer-wise state  $\mathcal{S}^l$  and an accuracy term  $\psi$ , from the environment. The environment’s



accuracy/precision  $\psi$  is computed with respect to the logits  $\tilde{Y}$  and labels of the validation set. Depending on the level of abstraction, the CNN  $f$  is simulated and scheduled on the HW-model  $\mu$ , returning estimates  $\varphi$  of an embedded application for the agent to produce a pruning action  $\mathcal{A}^l$ .



■ **Figure 1** Overview of HW-Flow enabling HW-model based pruning. The CNN environment (bottom) is pruned by a DDPG agent (right). The distinction of layer-wise sparsity is based on the three proposed HW-modeling abstraction levels - *Coarse*, *Mid* and *Fine*, which estimate the complexity of CNN workloads.

## 4.1 Problem Formulation

Without loss of generality, in an  $L$ -layer CNN, the convolutional layer  $l \in \{1, \dots, L\}$  receives an input feature map  $A^{l-1} \in \mathbb{R}^{H_i \times W_i \times C_i}$ , where  $H_i$ ,  $W_i$ , and  $C_i$  indicate the spatial height, width, and input channels respectively.  $A^0$  is the input image  $I$  to the CNN, as shown in Figure 1 (bottom). The weights  $W \in \mathbb{R}^{K_h \times K_w \times C_i \times C_o}$  are the trainable parameters of the individual layers, here  $K_h$ ,  $K_w$ , and  $C_o$  are the kernel dimensions and the number of output channels (filters) respectively. The input  $A^{l-1}$  is convolved with the weights  $W^l$ , where the kernels are moved over the input with stride  $s$ . In detail, the task of the agent  $\pi$  is to prune the input channels  $C_i$  of the environment by zeroizing the binary pruning mask  $\mathcal{A}^l = \{0, 1\}^{1 \times 1 \times C_i \times 1}$ . To select the most salient channels, the Hadamard product  $\odot$  is applied, giving a sparse representation  $\tilde{W}^l \in \mathbb{R}^{K_h \times K_w \times C_i \times C_o} = W^l \odot \mathcal{A}^l$ . Referring to Figure 1, zeroizing an input channel in the  $l$  layer will zero out the corresponding output feature map from the  $l-1$  layer. Consequently, the kernels of all filters in the  $l-1$  layer are also zeroed-out. Channel-wise pruning removes several weights from the CNN at once, causing a significant loss in accuracy. To mitigate this negative effect and guarantee an energy and latency efficient compression, the learning-based agent  $\pi$  has to learn good actions  $\mathcal{A}^l$ . The goal of HW-Flow is to complement well-established proxies, such as **OPs** and **Params** count, with more elaborate HW-model based estimates, which are conducive to finding efficient CNNs for embedded applications.



## 4.2 Deep Deterministic Policy Gradient-based Agent

The DDPG agent’s architecture, including the actor and the critic, is adopted from He et al. [16]. The agent is augmented with key rewards and state information, allowing it to understand the influence of its pruning actions on the inference hardware with respect to the energy estimates  $\varphi_E$  and the latency estimates  $\varphi_L$ , elaborated in Section 4.4. The newly adapted state  $\mathcal{S}$  is composed of the following layer information of the environment’s  $f$ : the index of the layer  $l$ , stride  $s$  and the layer dimensions after pruning  $\tilde{C}_o, \tilde{C}_i, W_i, H_i$ . It should be noted that the multi-level estimates  $\varphi$  obtained from the HW-models are considered to be part of the state  $\mathcal{S}$ , where  $\varphi = [\varphi^0, \dots, \varphi^l, \dots, \varphi^L]$  ensembles either layer-wise energy estimates  $\varphi_E$  or latency estimates  $\varphi_L$ . As expressed in equation 2, the action  $\mathcal{A}^{l-1}$  is applied for composing the input state of the agent.

$$\mathcal{S}^l = \langle l, s, \tilde{C}_o, \tilde{C}_i, W_i, H, \varphi^l, \sum_{i=0}^{l-1} \varphi^i, \sum_{j=l+1}^L \varphi^j, \mathcal{A}^{l-1} \rangle \quad (2)$$

In this work, the agent is trained using one of the two search protocols, either the estimate *balanced* or estimate *constrained* reward function, as defined in equation 3. The *balanced* reward of equation 3 is inspired by [17]. When the HW-constraints are unknown in the early stages of the design, the reward function can be formulated to achieve at least a target accuracy  $\psi^*$  before optimizing the performance estimate term. A trade-off between the accuracy term  $(1 - (\psi^* - \psi)/b)$  and the estimate term  $\log(\varphi^*/\varphi)$  after each pruning action is the goal of the estimate *balanced* reward function. Parameter  $b$  influences the turning point between a negative and positive reward  $\mathcal{R}$ , encouraging the agent to improve the accuracy when the difference between  $\psi^*$  and  $\psi$  is larger than  $b$ . When this condition is met, the agent starts to optimize the trade-off between accuracy and hardware estimates. This reward can also be extended to optimize multiple KPI’s by appending several logarithmic terms. The estimate *constrained* compression improves the reward  $\mathcal{R}$  by maintaining higher prediction accuracy  $\psi$  after each pruning action. This encourages the agent to prune the CNN model while minimizing the accuracy degradation when the HW-constraints are strictly stipulated.

$$\mathcal{R} = \begin{cases} (1 - \frac{\psi^* - \psi}{b}) \cdot \log(\frac{\varphi^*}{\varphi}), & \text{if } \textit{balanced} \\ \psi, & \text{otherwise } \textit{constrained} \end{cases} \quad (3)$$

The estimate term in the reward represents the benefits obtained from pruning with respect to the HW-model  $\mu$ , giving the estimate  $\varphi^*$  of the unpruned base model and  $\varphi$  after each episode of the agent.

## 4.3 HW-model Abstraction Levels

The HW-Flow framework proposes 3 levels of abstraction, *Coarse*, *Mid*, and *Fine*, for estimating the complexity of a CNN workload (Table 2). At the *Coarse* level, the goal would be to narrow down the CNN architectures which would suit an application’s accuracy requirements, while maintaining a reasonable number of OPs and Params.

Once satisfied with the CNN’s computational complexity, the *Mid*-level estimates take intermediate design parameters such as memory hierarchy, partitioning, and bandwidth into consideration, which decide whether the off-chip to on-chip communication infrastructure is suitable for the considered HW design. At the *Mid*-level, HW-Flow provides refined metrics such as off-chip to on-chip data transfer volumes, computation-to-communication ratio (CTC) and off-chip energy  $\varphi_{E, \text{off-chip}}$  due to external memory accesses. Using this information, the candidate CNNs can be tested for various criteria, such as respecting the communication bandwidth constraints expected

■ **Table 2** Input, output and optimization details of HW-Flow’s abstraction levels.

Level	Input	Optimization	Output
<b>Coarse:</b>	• CNN graph	• N/A	• OPs/Params
<b>Mid:</b>	<ul style="list-style-type: none"> <li>• Memory hierarchy</li> <li>• Memory size/partitioning</li> <li>• Off-chip bandwidth/burst length</li> <li>• Compute array sizes</li> </ul>	<ul style="list-style-type: none"> <li>• Loop tiling</li> <li>• Loop reordering</li> </ul>	<ul style="list-style-type: none"> <li>• CTC ratio</li> <li>• Off-chip energy/accesses: <math>\varphi_{E,off-chip}</math></li> <li>• Mid Latency <math>\varphi_{L,bandwidth}</math></li> </ul>
<b>Fine:</b>	<ul style="list-style-type: none"> <li>• Complete memory/compute hierarchy</li> <li>• PE specification</li> <li>• Supported dataflows</li> </ul>	<ul style="list-style-type: none"> <li>• Loop unrolling</li> <li>• Interleaving</li> <li>• Folding</li> <li>• Mapping exploration</li> </ul>	<ul style="list-style-type: none"> <li>• Total inference energy: <math>\varphi_{E,NE}</math></li> <li>• Breakdown of datatype energy</li> <li>• Total inference latency: <math>\varphi_{L,inference}</math></li> <li>• Detailed data movement schedule</li> </ul>

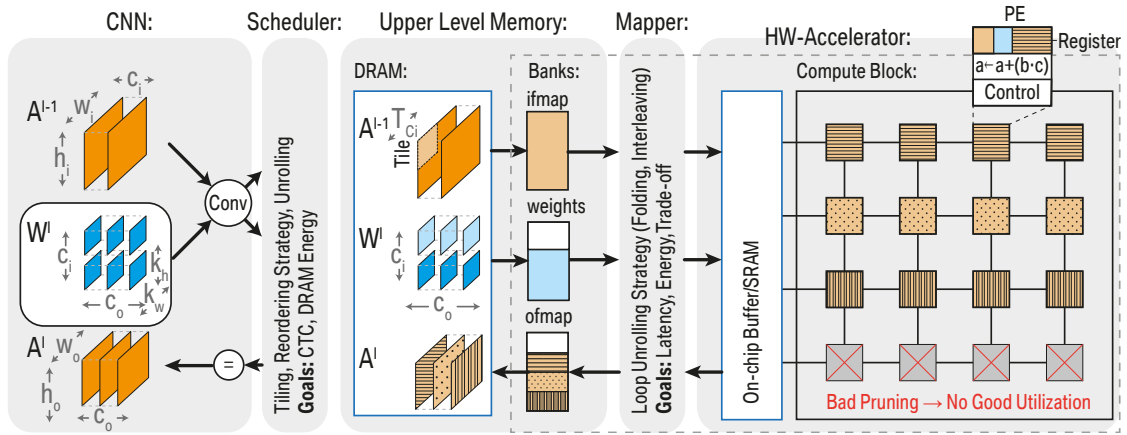
on the hardware platform, resulting in latency estimates  $\varphi_{L,bandwidth}$  at the *Mid* level. In Section 5, we demonstrate how the *Mid*-level can provide an intermediate evaluation closer to the detailed *Fine*-level without requiring the designer to decide the low-level details of the compute array at this stage. This eases the design process into the next stage and provides one more stepping-stone before narrowing down the complete HW design.

HW-Flow’s *Fine* estimates provide metrics that consider specific details of the accelerator, such as the detailed structure and dimensioning of the processing element (PE) array and the scheduling strategies supported by the on-chip interconnect, which determines the possible communication channels between the PEs. The estimates at this level provide normalized energy costs  $\varphi_{E,NE}$  of each datatype  $dtype \in \{\text{ifmap}, \text{ofmap}, \text{psum}, \text{weight}\}$  at each memory level of the system. The agent  $\pi$  receives estimates for the energy and latency for a particular scheduling strategy relative to each layer of the investigated network. At the *Mid* and *Fine* levels, HW-Flow additionally searches for scheduling solutions which optimize the criteria provided by the designer, e.g. latency, energy or a trade-off. The latency is the time interval between the stimulation of the host and its response from the accelerator for a particular neural network, i.e. time between the pre-processed input and the output of the neural network. The total energy consumption of the CNN accelerator is determined using the data movement cost at various memory hierarchies and compute cost in processing element array.

For a human designer, the three levels allow the structured traversal of the HW and CNN design space, which may be a daunting task otherwise. The designer starts with a set of potential CNNs and tests their pruning potential in terms of OPs, Params and task-related accuracy. Once a subset of CNNs with high pruning ratios and acceptable task-related accuracies is narrowed down, the *Mid*-level is then critical in designing the communication infrastructure between the host and the accelerator, as well as dimensioning the on-chip SRAM. After pruning, an under-dimensioned SRAM can result in high communication effort with off-chip DRAM, resulting in higher stress on the off-chip to on-chip interconnect. An over-dimensioned SRAM can lead to area-on-chip and fabrication cost problems. Therefore, the pruning potential of the CNN needs to be evaluated *alongside* the on-chip SRAM dimensions and the interconnect stress at the *Mid*-level. Finally, at the *Fine*-level, the designer can use the findings at the *Coarse* and *Mid*-level and further specify the computation infrastructure of the accelerator (such as PE count, register files sizes, dataflow). Tackling all three levels by searching for an efficient HW and CNN configuration at once would potentially lead to sub-optimal results. This would also result in more GPU hours of CNN pruning and HW search due to the difficulty in understanding which part of the system (CNN or HW) needs to be adjusted to meet the demands of the application at hand.

#### 4.4 HW-model Optimizer

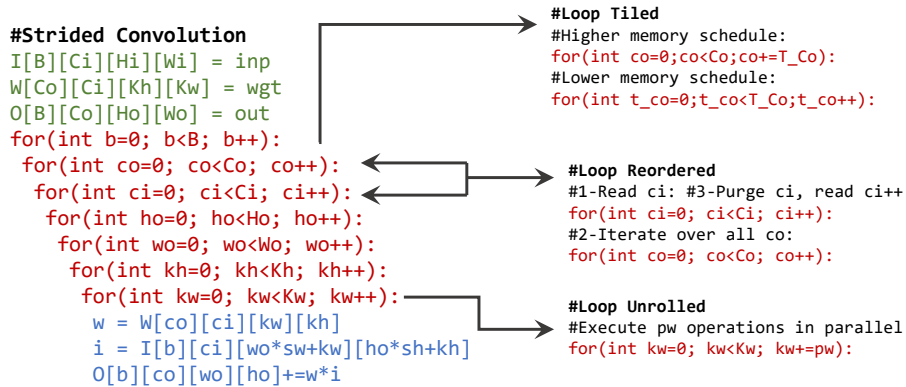
**Model:** The core structure of the HW-model in HW-Flow is based on two generic blocks, namely the memory and compute blocks. Arbitrary memory hierarchies can be instantiated using these generic blocks. Each block is accounted for its position in the hierarchy by referring the memory below it and the level at which it is placed. The highest level represents the largest memory, where all the data fits. Memory blocks can be detailed with their total or datatype-wise segmented size. Below last-level memories, a compute block can be instantiated, as shown in Figure 2. The compute block is defined by several parameters, including the number of processing elements (PE), interconnect dimensions, and register file sizes. The register files in each PE can be specified



■ **Figure 2** Scheduling strategies for data movement optimization.

according to their size and segmentation, as shown in Figure 2 (top/right). Using these blocks, diverse compute architectures can be described. In this paper, we focus on modeling architectures similar to [4], with a single on-chip buffer and a compute core with an array of PEs, as depicted in Figure 2.

**Scheduler.** The energy contribution of data movement cannot be disregarded for efficient execution of CNNs. For most cases, it constitutes the majority of the total power consumption to execute CNN models. CNNs are commonly represented in a nested loop format, as expressed in Figure 3. The for-loops shown present many reuse opportunities. The main computation



■ **Figure 3** Nested for-loop representation of strided convolution.

is at the core of the inner-most loop and many elements are accessed in multiple iterations of the higher loops. Specifically, reuse occurs when the indices of the parameters involved in the inner-most computation remain fixed for some loops before iterating in others. In hardware, this translates to a single element being stored at a lower level memory for multiple iterations before being purged to make space for new data. For optimal reuse to occur, no single element should be read more than once from a higher level memory. This implies that during all the iterations that a single element is involved in, all the other elements that it is reused against also fit in the lower level memory. Practically, due to memory constraints, the parameters required by the entire nested-loop do not fit in the lowest-level of the memory hierarchy. A standard method of exploiting the entire hierarchy is to relax this constraint and split the for-loops into shallower loops through a technique called *loop-tiling*. As shown in Figure 2, the loop tiling strategy effectively decides which tiles  $T \in \{T_{C_i}, T_{C_o}, T_{H_o}, T_{W_o}, T_{K_w}, T_{K_h}, T_B\}$  of CNN computation will take place in one round of communication with a lower level memory. Note that  $T_B$  is the tiling along the batch dimension when performing batch processing. The tiling strategy is selected based on the amount of on-chip buffer  $Buf$ , respecting the inequality in equation 4.

$$\underbrace{T_{H_i} \times T_{W_i} \times T_{C_i} \times T_B}_{\text{Input Tile}} + \underbrace{T_{H_o} \times T_{W_o} \times T_{C_o} \times T_B}_{\text{Output Tile}} + \underbrace{K_h \times K_w \times T_{C_i} \times T_{C_o}}_{\text{Weight Tile}} \leq Buf \quad (4)$$

To generate an output tile with spatial dimensions  $T_{W_o}, T_{H_o}$  with stride  $s$  and kernel  $k$ , an input tile with spatial dimensions  $T_{W_i}, T_{H_i}$  are required. The relation between input and output tiles is given in equation 5.

$$\begin{aligned} T_{W_i} &= (T_{W_o} - 1) \cdot s + K_w \\ T_{H_i} &= (T_{H_o} - 1) \cdot s + K_h \end{aligned} \quad (5)$$

The order of the loops can also be manipulated dynamically for each layer without affecting the algorithm through *loop-reordering*. As an example in Figure 3, loop  $C_i$  can be swapped with loop  $C_o$ , allowing a single element  $c_i$  to reside longer on the lower-level memory while iterating over all possible elements  $c_o \in C_o$ . This can help extract improved reuse opportunities since the lower-level loops remain on the lower-level memories of the hardware architecture, thus closer to the compute units. Particularly for *Mid*-level estimates, these permutations directly impact the number of DRAM accesses and consequently DRAM energy. This work considers three loop orders, namely Input Reuse Order (IRO), Weight Reuse Order (WRO), and Output Reuse Order (ORO) schemes inspired by the work in [35]. Switching dynamically between these three reuse schemes allows to schedule the entire CNN exploiting the reuse opportunities of different layers. As an example, layers with a very large kernel can benefit from ORO and WRO schedules, whereas layers with large feature maps (e.g. the first layers of most conventional CNN) will benefit the most with IRO schedule. This is referred to as dynamic loop tiling. Finally, once a memory level is distributed spatially, further loops can be unrolled over the parallelism degree offered by the hardware architecture through *loop-unrolling*. In Figure 3, the kernel's elements can be assigned to spatially distributed processing elements, executing several  $K_w$  loop iterations in parallel during a single clock cycle. HW-Flow's mapper component optimizes the execution schedule through this technique, as detailed in the following subsection.

Using the aforementioned strategies, the scheduler builds a matrix of possible tilings and loop orders. Each potential solution in the matrix is checked for *legality*, by assessing whether its transfer size breaches the memory restrictions at lower levels. The volumes  $\mathcal{V}$  moved between the memory levels are calculated based on the memory occupation and the number of invocations required. To analytically reduce the size of the search space, a Computation-to-Communication (CTC) *hall-of-fame* is constructed after the evaluation of all legal solutions, which contains only

a top percentage of the highest CTC loop tilings/orderings. Equation 6 represents a CTC ratio formula, inspired by the work in [44].  $\gamma$  represents a bandwidth-correction term to account for the burst-length of the memory transfers. The numerator is the number of operations/complexity of a particular workload. The denominator is the overall DRAM access along with bandwidth scaling for input, weights, and outputs for a particular workload.

$$\text{CTC} = \frac{2 \cdot H_o \cdot W_o \cdot K_h \cdot K_w \cdot C_o \cdot C_i}{\sum_{\text{dtype}} \gamma_{\text{dtype}} \cdot \mathcal{V}_{\text{dtype}}},$$

$$\text{dtype} \in \{\text{ifmap}, \text{ofmap}, \text{psum}, \text{weight}\} \quad (6)$$

The hall-of-fame solutions are passed on to the mapper for *Fine*-level estimations. An analysis on the hall-of-fame size and the efficiency of the final schedule produced is presented in Section 5.4.

**Mapper.** Many dataflow strategies have been explored in literature [4, 43]. Reuse opportunities in CNNs include convolutional, weight, input, and partial sum reuse. In this work, we focus on three dataflows, namely weight-stationary, output-stationary, and row-stationary. The weight-stationary dataflow unrolls the dimensions  $T_{C_i}$  and  $T_{C_o}$  as  $P_{C_i}$  and  $P_{C_o}$  across the spatially distributed computation array. Each PE holds complete kernels ( $K_w \times K_h$ ) and corresponding input feature slices. Spatial reduction of partial sums can occur inside the PE; however, accumulation across input channels requires `psum` traversal over the spatial computation array. The output-stationary dataflow similarly unrolls  $P_{C_i}$  and  $P_{C_o}$ ; however, the `psums` remain stationary in each processing element, while input feature map pixels traverse the array and kernel pixels are updated once they are exhaustively used over the tile. Finally, row-stationary as introduced in [4] unrolls the  $T_{H_o}$  dimension horizontally across the array as  $P_{H_o}$ . Each  $K_h$  column of PEs is responsible for the complete computation of an entire row of the output  $W_o$ , while the neighboring set of  $K_h$  PEs computes the output row below that. Folding and replication techniques are applied to fit this unrolling method on the physical array dimensions. All three dataflows enable interleaving of channel computation within a single PE to maximize the use of the register files.

HW-Flow’s mapper analytically determines the viability of a particular dataflow, based on the hardware details such as the interconnect dimensions, processing array size, and scratchpad configuration. HW-Flow attempts to find a mapping that optimizes a given criterion (energy, latency, or a trade-off) while respecting the dataflow’s restrictions. As presented in Figure 3, unrolling a subset of a loop’s iterations as  $P$  spatially distributed computations, improves execution time. Assuming a filled pipeline, the latency of a layer  $\varphi_L$  can be estimated as the product of intertile and intratile latency as shown in equation 7. The intertile latency is computed based on the number of tiles required to transfer from off-chip memory to on-chip memory. Based on the PE unrolling procedure of the tiles available in the on-chip memory, the intratile latency is calculated. In equation 7, the kernel dimensions  $K_h$  and  $K_w$  are not tiled, as such granular tilings result in performance degradation for modern CNN models with small kernel sizes.

$$\tilde{\varphi}_{L,\text{interTile}} = \left\lceil \frac{C_o}{T_{C_o}} \right\rceil \cdot \left\lceil \frac{C_i}{T_{C_i}} \right\rceil \cdot \left\lceil \frac{H_o}{T_{H_o}} \right\rceil \cdot \left\lceil \frac{W_o}{T_{W_o}} \right\rceil$$

$$\tilde{\varphi}_{L,\text{intraTile}} = \left\lceil \frac{T_{C_o}}{P_{C_o}} \right\rceil \cdot \left\lceil \frac{T_{C_i}}{P_{C_i}} \right\rceil \cdot \left\lceil \frac{T_{H_o}}{P_{H_o}} \right\rceil \cdot \left\lceil \frac{T_{W_o}}{P_{W_o}} \right\rceil \cdot \left\lceil \frac{T_{K_h}}{P_{K_h}} \right\rceil \cdot \left\lceil \frac{T_{K_w}}{P_{K_w}} \right\rceil \quad (7)$$

$$\tilde{\varphi}_{L,\text{total}} = \tilde{\varphi}_{L,\text{interTile}} \times \tilde{\varphi}_{L,\text{intraTile}}$$

A particular mapping produces reuse factors for each datatype at different memory levels. We denote a reuse factor with  $R_{\text{level}}^{\text{dtype}}$ , where  $\text{level} \in \{\text{Offchip}, \text{Onchip}, \text{Array}, \text{Registers}\}$ . Reuse factors are dependent on tiling and unrolling strategies, as well as data interleaving [4], where

a single computation element switches between multiple sets of the same datatype in order to extend the utilization of its registers. Once a legal mapping is found, the energy contributions of each datatype at each memory level can be computed. Equation 8 shows an example of the energy consumption calculation at a particular memory level for a single datatype [4]. The read/write cost term  $\mathcal{C}$  of a particular memory level can be set based on the fabrication technology or a relative normalized cost to other memory types in the hardware architecture. The energy estimates of all datatypes at all memory levels can be calculated similarly and summed up to obtain the total layer energy  $\varphi_E$ .

$$\varphi_{E,\text{Level}}(\text{dtype}) = \left( \prod_{\text{off-chip}}^{\text{Level}} R_{\text{level}}^{\text{dtype}} \right) \cdot \mathcal{C}_{\text{Level}}$$

$$\forall \text{ dtype} \in \{\text{ifmap, ofmap, psum, weight}\} \quad (8)$$

Finally, the mapping found is fed back to the scheduler, determining whether the tiling factors it provided were adequate. The possible combinations for legal schedules are evaluated and compared. These two optimization problems are codependent, as a tiling strategy that optimizes off-chip data movement may result in a mapping that under-utilizes the processing elements for a particular dataflow and vice versa. Therefore, a feedback loop, such as the one in HW-Flow, is essential in finding an optimal scheduling strategy for the overall system.

## 4.5 Search Space for HW-model Optimizer

For *Fine*-level estimates, creating a complete schedule implies choosing a fixed set of tiling factors  $\{T_{C_i}, T_{C_o}, T_{H_o}, T_{W_o}\}$  and unrolling factors  $\{P_{C_i}, P_{C_o}, P_{H_o}, P_{W_o}, P_{K_h}, P_{K_w}\}$ . We restrict  $T_{K_h} = K_h$  and  $T_{K_w} = K_w$ , and therefore omit them from the tiling factors set. Modern CNNs employ small kernel sizes, making it unreasonable to tile them during computation. Furthermore, tiling the kernel dimension generates a large amount of partial sums which can quickly become *parasitic* due to memory consumption and on-chip movement, if not collapsed into an output pixel. We can define two subspaces in the scheduling search space: tiling space  $\mathcal{T}$  and mapping space  $\mathcal{P}$ . Equation 9 defines the size of the subspaces. *Ord* defines the reordering possibilities of the outer (off-chip memory) loops of the convolution. In this work, we consider three distinct orderings, IRO, ORO, and WRO, relating to inputs, outputs, or weights being kept longer on the on-chip memory respectively [35].

$$|\mathcal{T}| = C_i \cdot C_o \cdot H_o \cdot W_o \cdot \text{Ord}$$

$$|\mathcal{P}_\tau| = T_{C_i} \cdot T_{C_o} \cdot T_{H_o} \cdot T_{W_o} \cdot T_{K_h} \cdot T_{K_w} \quad \forall \tau \in \mathcal{T} \quad (9)$$

$|\mathcal{T}|$  and  $|\mathcal{P}_\tau|$  represent the cardinality of the tiling space and mapping space associated with a single tiling  $\tau \in \mathcal{T}$  respectively. Therefore, the size of  $\mathcal{P}_\tau$  is directly dependent on a single solution  $\tau = \{T_{C_i}, T_{C_o}, T_{H_o}, T_{W_o}, \text{Ord}\} \in \mathcal{T}$ . Restricting  $\mathcal{T}$  directly reduces the number of total  $\mathcal{P}_\tau$  searches necessary for finding a schedule.  $\mathcal{T}$  may contain a single solution  $\tau$  which results in a single mapping  $\rho \in \mathcal{P}_\tau$ , that is optimal for the overall schedule, in terms of latency, energy, or both. The trade-off in restricting the size of  $\mathcal{T}$  is between schedule search speed and the optimality of the found schedule. To avoid evaluating drastically sub-optimal tilings, we analytically reduce the size of  $\mathcal{T}$ , and maintain solutions  $\tau$ , which have a higher probability of producing efficient  $\rho$  mappings. The search for the optimal mapping  $\rho$  can also be expedited with further sampling techniques.

A straightforward approach to restricting the search space is to uniformly sample equidistant solutions in  $\mathcal{T}$ . We choose uniform sampling over random sampling to consider, at a minimum, a single candidate from each neighborhood in the search space. For fixed dimensions  $C_i, C_o, H_o,$

and  $W_o$ , the distance between two solutions depends on the sampling step. For small search spaces, the sampling step can be set to a small integer value. Therefore, for all experiments on the CIFAR-10 dataset, the sampling step was set to 2, effectively halving the number of tiles from each dimension. For the larger CNN models, better suited for the ImageNet dataset, integer steps are less effective. The size of a particular dimension  $C_i$ ,  $C_o$ ,  $H_o$ , and  $W_o$ , varies greatly between the first layer of the CNN towards the last. This makes the choice of a single integer step-size for all dimensions either grossly large to maintain simulation speed or small to maintain optimality at the cost of prohibitively increased search time. We use a ratio-based sampling to overcome this problem, where the step-size is a fixed fraction of the total dimension. This decouples the dimensions of the CNN from the number of  $\tau$  mappings to be evaluated. We also allow each dimension to have its own ratio, providing more flexibility in finely searching smaller dimensions and coarsely searching larger ones. One more technique to aggressively reduce the search space is to find all the factors (divisors) of a particular dimension and declare those as the possible tiling factors. Since a factor will always give an integer number of tiles, this method usually leads to near-optimal results and is scalable to larger CNNs.

The CTC ratio metric is elaborated in Section 4.4 for choosing a reasonable tiling solution. Based on the intuition that a high CTC tiling solution  $\tau$  could result in an efficient mapping, we analytically reduce the search space by creating a CTC hall-of-fame (HOF). In the first step, we evaluate the CTC ratio for all  $\tau \in \mathcal{T}$ , which is a fast and parallelizable operation. A set percentage of  $\mathcal{T}$  with the highest CTC ratios among all the solutions is entered in the HOF. Only members of the HOF have their respective  $\mathcal{P}$  searched for mapping solutions.

For *Mid*-level estimates, evaluating the outputs shown in Table 2 is a fast and parallel operation, which can be done by either sampling the tiling space or exhaustively, since the total number of solutions to evaluate at this level is  $|\mathcal{T}|$ . For *Fine*-level, we have a total number of full schedule evaluations equal to  $\sum_{\tau} |\mathcal{P}_{\tau}|$ , which rapidly grows with  $\mathcal{T}$ , emphasizing the importance of good search space reduction techniques to maintain reasonable search time, without cutting out the optimal solutions in the space.

## 5 HW-model Design Space Exploration

We evaluate the HW-Flow framework by exploring the HW estimations at various abstraction levels in Section 5.1. We explore the design choices and estimates at the *Mid* and *Fine*-level abstractions in Section 5.2 and Section 5.3 respectively. We improve the schedule search time of HW-Flow by systematically reducing the search space of the proposed HW-model optimizer using a detailed ablation study in Section 5.4. Finally, in Section 5.5, the optimal mapping is validated with the estimates reported in Eyeriss [4] and compared against the HW modeling framework Timeloop [27]. An advantage of using HW-models over HIL-based methods is the flexibility of prototyping and testing multiple target architectures before committing to a final design for synthesis and fabrication. We report various HW configurations with different PE array sizes, memory costs, SRAM buffer and register sizes in Table 3. Column 3 indicates the data access cost from higher memory levels (DRAM) to lower levels (RF) relative to one MAC operation. This section uses the HW-Flow-Val model with 16-bit word length to explore the *Coarse*, *Mid* and *Fine* abstraction levels and validate the modeling tool.

### 5.1 HW Estimations at Various Abstraction Levels

In this subsection, we demonstrate the HW estimates produced across various abstraction levels and discuss their use in the context of HW development. For this purpose, we interpret the influence of on-chip buffer size through DRAM access counts and throughput of the HW-Flow-Val



■ **Table 3** Hardware configurations used for experiments and validation. RS refers to row-stationary dataflow.

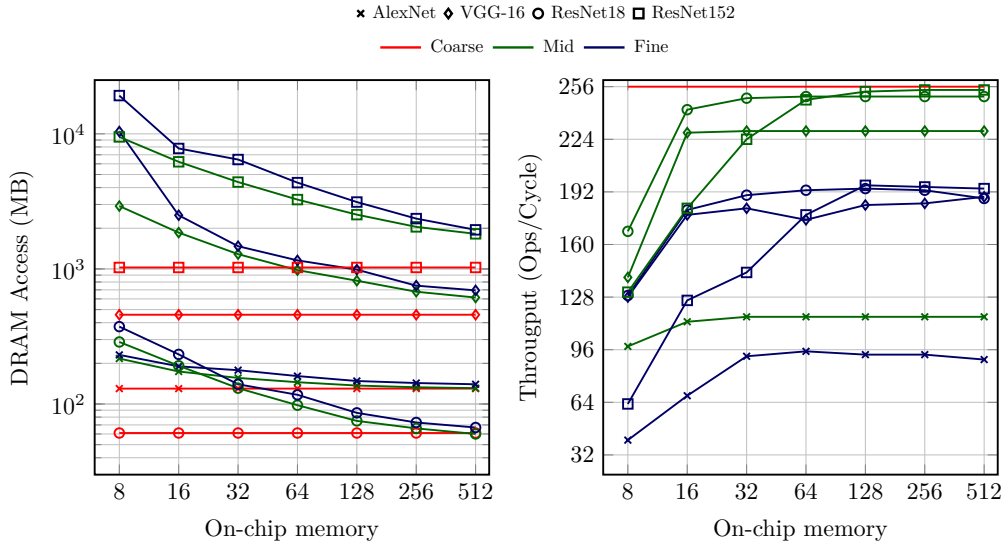
Hardware Model	Architecture Spec	PE Array	Memory Cost DRAM, SRAM, Array, RF	SRAM size <KB>	Register Words filter, ifmap, psums
<b>HW-Flow - Val</b>		16 × 16	200, 6, 2, 1	128	192, 12, 16
<b>Timeloop [27]</b>		16 × 16	200, 7.41, 0, 1	128	192, 12, 16
<b>Eyeriss-like-168 PE (RS)</b>		12 × 14	200, 6, 2, 1	128	224, 12, 14
<b>Eyeriss-like-256 PE (RS)</b>		16 × 16	200, 13.84, 2, 1	256	224, 12, 14
<b>Eyeriss-like-1024 PE (RS)</b>		32 × 32	200, 155.35, 2,1	3072	224, 12, 14
<b>Eyeriss-like-Deeplab (RS)</b>		32 × 32	200, 155.35, 2,1	3072	224, 37, 16

model in Figure 4. We obtain the *Coarse*-level estimations for DRAM access counts (indicated in red) by simply summing-up the layer-wise transfer volumes of `ifmaps`, `ofmaps` and `weights`. This is equivalent to considering that the HW has unbounded buffers and communication bandwidth. Similarly, we consider that all the compute units in the PE array are fully occupied and report the accelerator’s throughput. These assumptions in the initial phases of development allow the designer to choose the CNN topologies that suit the application under consideration. For *Mid*-level estimations, we optimize layer-wise schedules for minimum DRAM energy by considering the possibility of dynamic tiling and reordering schemes as described in Section 4.4. We report the sum of DRAM accesses across all layers (indicated in green). To calculate the accelerator’s throughput at *Mid*-level, we consider the external memory bandwidth as 8 words/cycle. For *Fine*-level estimations, we optimize the row-stationary dataflow to obtain a trade-off between normalized energy and latency (indicated in blue).

We perform the measurements for four CNN architectures, namely AlexNet, VGG-16, ResNet18, and ResNet152. We observe that as the on-chip buffer size increases, larger tiles of input, weights, and outputs can be stored on the buffer, thereby decreasing the number of DRAM Accesses. We also observe that the *Mid* and *Fine* estimations for all the CNN architectures could meet the ideal *Coarse* estimations at higher buffer sizes ( $\geq 512\text{KB}$ ). We notice that the *Fine*-level estimations produce a higher number of DRAM accesses, as the schedule must consider more complex HW details and constraints at this level. The AlexNet architecture achieves the least throughput among other architectures, as a considerable number of operations and parameters are assigned to fully-connected layers. The throughput produced at the *Fine*-level considers the dataflow and the underlying unrolling scheme and therefore achieves closer estimates compared to real target deployment. We observe that the throughput saturates at 128KB of buffer size for both *Mid* and *Fine* level estimations for different CNN architectures. A slight decrease in throughput happens for AlexNet and ResNet18 at *Fine*-level scheduling for on-chip memories larger than 128KB. This due to the *Fine*-level schedule simultaneously optimizing for inference energy (not shown in the figure) as the on-chip memory grows.

## 5.2 Mid-Level Estimations and Design Choices

In Figure 5, we evaluate the number of DRAM accesses for different loop ordering schemes. We also evaluate the *Mid*-level throughput estimation for different external bandwidth considerations (4, 8, 16 words/cycle). Among IRO, WRO and ORO loop ordering schemes, we observe that the IRO scheme produces DRAM accesses close to the layer-wise dynamic ordering scheme for AlexNet. For ResNet18, we observe that the ORO scheme achieves DRAM access closer to the dynamic order. This emphasizes the importance of a dynamic ordering scheme as different workloads prefer reuse



■ **Figure 4** Analysis of DRAM Access and Throughput on varying the on-chip buffer size and different CNN architectures.

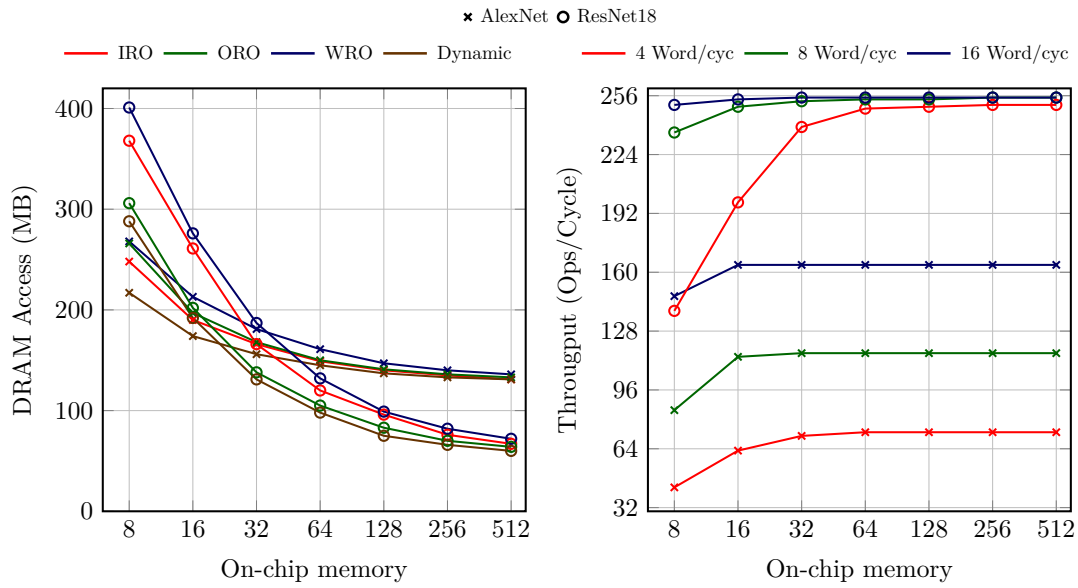
for different datatypes. Fixing the dynamic ordering scheme, the throughput of the accelerator is analyzed for AlexNet and ResNet-18 under different external memory bandwidth considerations. We observe a significant improvement in the throughput of AlexNet as the bandwidth increases. For ResNet18, the throughput saturates at 8 words/cycle. Improving the throughput for AlexNet depends on the choice of external memory bandwidth as AlexNet has several memory-bounded fully-connected layers compared to ResNet18.

### 5.3 Fine-Level Estimations and Dataflows

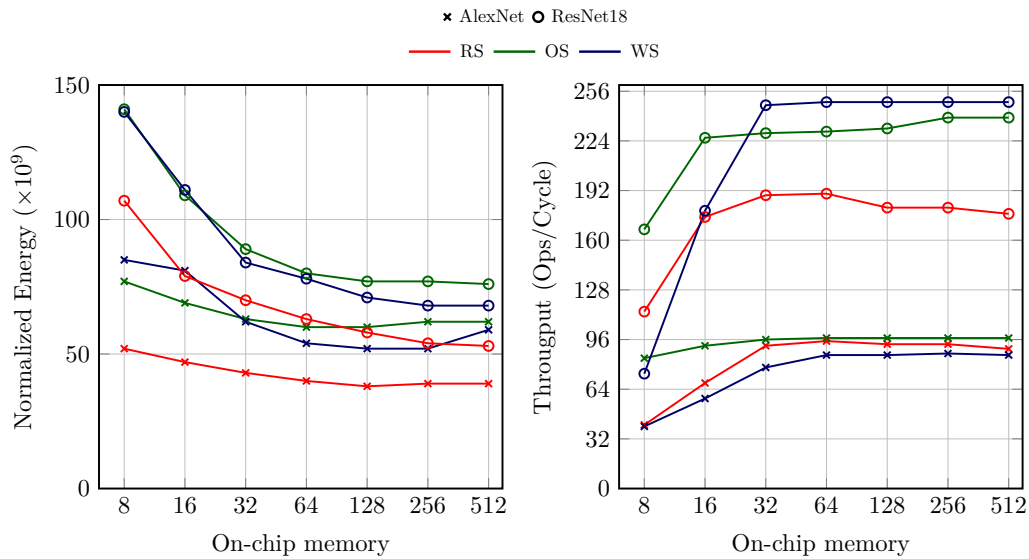
The row-stationary (RS), weight-stationary (WS) and output-stationary (OS) dataflows, detailed in the Section 4.4, are used to explore the *Fine*-level estimates. The mapper searches for a trade-off between normalized energy and latency while respecting each dataflow’s unrolling rules and the HW’s memory and compute capacity checks. We obtain the *Fine* estimates for AlexNet and ResNet18 models for different on-chip buffer sizes considering the HW-Flow-Val model. We observe that the RS dataflow is the most energy efficient at all the buffer sizes. This is due to RS maximizing the data reuse at the register-level, for all the datatypes [4]. OS and WS dataflows maximize the compute utilization of PE arrays, albeit with higher normalized energy requirements. The WS dataflow achieves higher throughput using larger buffer sizes ( $\geq 32KB$ ) for ResNet18.

### 5.4 Search Space Exploration for Fine-level Estimations

We perform multiple experiments to measure the sensitivity of the scheduling tool under the search space sampling strategies described in Section 4.5. Although all schedules produced under any of the sampling strategies are valid, it is favorable to maintain schedules which are close or comparable to the optimum for a particular CNN workload. To explore the search space, we use the HW-Flow-Val model with RS dataflow to estimate convolutional layers of the AlexNet model with 16-bit weights and activations, as it offers a diverse set of workloads with different kernel sizes and strides. The input batch size is set to 16. AlexNet consists of convolutional workloads with strides 4, 2 and 1 and kernels sizes 11, 5 and 3. Grouped convolution is performed for layers 2, 4 and 5.



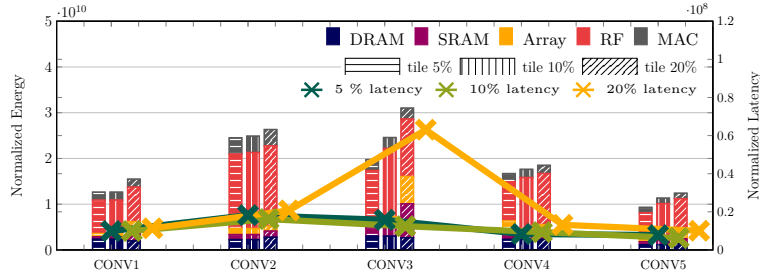
■ **Figure 5** Influence of loop reordering schemes and external memory bandwidth on DRAM access and throughput of the HW accelerator.



■ **Figure 6** Influence of dataflows selection on normalized energy and throughput of the HW accelerator.

Table 4 shows the search time needed for different analytical search strategies to produce a schedule. The quality of the search method can be measured by its corresponding mapping goal. Three different  $\mathcal{T}$  sampling rates (5%, 10%, 20%) are explored in Figure 7 with 1% CTC-HOF. We observe that the normalized energy increases as we limit the exploration by increasing the sampling rate. Based on the trade-off between evaluation speed (see Table 4) and schedule optimality (Figure 7), we sample the tile space with 10% for ImageNet experiments to obtain HW estimates

for the pruning process. This results in an overall shorter search time (up to  $\times 2.5$ ) at the cost of degradation in mapping optimization goal. We also highlight the tile sampling method by computing divisors in Table 4. We observe that the divisors-based sampling method produces a schedule with the lowest energy consumption. However, this method might produce sub-optimal results in case of channel pruning when the agent finds prime-number of filters for a particular layer. For CIFAR-10 experiments, we use smaller, integer steps of 2, as these CNNs have a small scheduling search space.



■ **Figure 7** Sensitivity analysis for search space sampling of tiling factors.

The sensitivity analysis of the CTC-HOF tile space reduction technique is shown in Table 4. The results show that the (10%  $\mathcal{T}$ , 1% HOF) strategy is very effective, providing a speedup of  $2.14\times$  compared to (10%  $\mathcal{T}$ , 100% HOF) schedule without sacrificing the optimality of the schedule. Combining these methods is critical in maintaining a reasonable exploration time for multiple pruning experiments. We finally use the sampling strategy (10%  $\mathcal{T}$ , 1% HOF) with an overall search time reduction of  $20\times$  compared to the search strategy (5%  $\mathcal{T}$ , 100% HOF). Once a HW-CNN pair is found, the HW-optimizer can run with a more exhaustive search strategy and provide an improved schedule for the final deployment stage. For reference, we also present the search/calculation time for *Mid* and *Coarse* estimate levels in Table 4. In addition to facilitating the proposed codesign approach, the two higher abstraction levels are much faster to estimate, allowing agent  $\pi$  to run for more episodes than with *Fine*-level estimates, for the same amount of time.

## 5.5 Validation with Eyeriss and Timeloop

**Validation:** To validate the correctness of HW-Flow’s modeling and mapping components, we compare its estimates with the Eyeriss architecture [4] and its Timeloop model [27] for AlexNet [21] inference, which has diversified kernel sizes, strides and input/output dimensions. Figure 8 shows a breakdown of normalized energy contributions of each datatype at each memory level for the convolutional layers. We observe that HW-Flow tracks the original Eyeriss results similar to Timeloop. A slight offset is observed, which can be attributed to small differences in the energy references used during the search. The overlapping line charts show the latency estimates of both frameworks.

## 6 Experimental Results

In Sections 6.1 to 6.6, we demonstrate the influence of channel pruning on different abstractions, reward functions, target HW architectures and mapping schemes using the estimates generated by HW-Flow. The pruning is evaluated based on CIFAR-10 [19] and ImageNet [30] for the classification task and CityScapes [5] for the semantic segmentation task. The 50K train and 10K

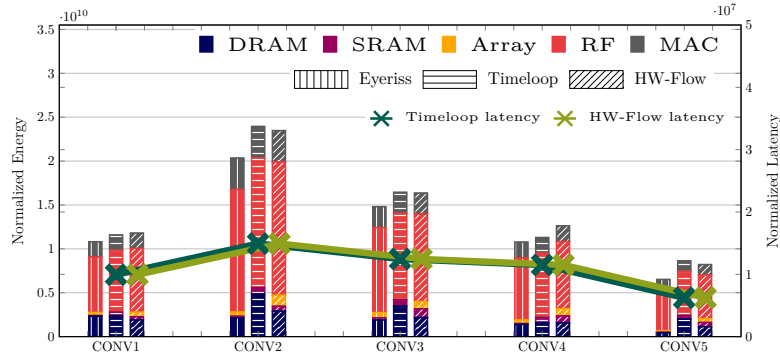
■ **Table 4** Schedule search duration and optimality under different search space reduction strategies for AlexNet on Eyeriss-like-256. All schedules optimize for a trade-off between latency and energy, unless marked otherwise. Similar to Eyeriss [4], we normalize DRAM energy (column 3) and total energy (column 4) to the cost of one MAC operation.

Search Strategy	Search Time [s]	DRAM Energy [ $\times 10^9$ ]	Energy [ $\times 10^9$ ]	Latency [ $\times 10^6$ cycles]
10% $\mathcal{T}$ , 1% HOF*	9.87	10.76	<b>83.25</b>	379
10% $\mathcal{T}$ , 1% HOF**	10.13	155.82	257.49	<b>65</b>
10% $\mathcal{T}$ , 1% HOF	10.23	11.06	91.89	67
5% $\mathcal{T}$ , 1% HOF	25.59	12.10	83.96	60
10% $\mathcal{T}$ , 1% HOF	10.23	11.06	91.89	67
20% $\mathcal{T}$ , 1% HOF	<b>7.23</b>	10.47	104.61	118
divisors $\mathcal{T}$ , 1% HOF	12.86	14.60	<b>71.61</b>	65
5% $\mathcal{T}$ , 100% HOF	215.42	12.10	83.96	<b>60</b>
5% $\mathcal{T}$ , 1% HOF	25.59	12.10	83.96	60
10% $\mathcal{T}$ , 100% HOF	23.03	11.06	91.89	67
10% $\mathcal{T}$ , 10% HOF	15.24	11.06	91.89	67
10% $\mathcal{T}$ , 1% HOF	<b>10.76</b>	11.06	91.89	67
divisors $\mathcal{T}$ , 100% HOF	51.47	9.67	<b>72.44</b>	65
divisors $\mathcal{T}$ , 1% HOF	12.86	14.60	71.61	65
Coarse Estimates	0.10	-	-	-
Mid Estimates 5% $\mathcal{T}$ ***	5.23	<b>8.13</b>	-	-

All simulations were run with 24 threads on an Intel Xeon E5-2698 Process  
Mapping goal : \*energy, \*\*latency, \*\*\*dram access

test images of CIFAR-10 are used to train and evaluate the base models, respectively. The images have a resolution of  $32 \times 32$  pixels. ImageNet consists of  $\sim 1.28$ M train and 50K validation images with a resolution of  $256 \times 256$  pixels. The CityScapes dataset consists of 2975 training images and 500 validation images, including their corresponding ground truth labels. The images of size  $2048 \times 1024$  show German street scenes along with their pixel-level semantic labels of 19 classes. The pruning experiments are performed using the agent detailed in Section 4.2, for 150 episodes of pruning exploration and 400 for learning and exploitation. After the agent selects the best action corresponding to the reward, the environment is fine-tuned for 60 epochs with a learning rate of  $1e-03$  for CIFAR-10 experiments. For ImageNet experiments, we fine-tune for 20 epochs with an initial learning rate of  $1e-02$ , step decay of  $1e-01$  for every 5 epochs. For CityScapes experiments, we fine-tune the model with a learning rate of  $1e-02$  for 240 epochs with a polynomial learning rate. If not otherwise mentioned, all hyper-parameters specifying the task-related training were adopted from the CNN’s base model. The batch size is set to 4 to evaluate the HW estimates.

HW metrics such as DRAM accesses, normalized energy, and latency are generated based on the different variants of an Eyeriss-like architecture [4] mentioned in Table. 3. These metrics are also reported similarly in the works of Timeloop [27] and Eyeriss [4]. In Table 5-10, the reported normalized energy estimates are relative to the cost of one MAC operation and are therefore unitless. The latency is reported as number of clock cycles. Additionally, we report the accuracy and pruning rate for each experiment. The pruning rate indicates the number of operations reduced relative to the baseline CNN. For comparison with the state of the art, we measure the memory required to store training parameters under 16-bit fixed-point representation. To demonstrate the effect of pruning on different HW-architectures, we scale the variants of spatial, eyeriss-like accelerators from 168 to 256 and 1024 PEs (Table 3).



■ **Figure 8** Validation with the Eyeriss accelerator [4] and Timeloop [27]. Note: [4] does not report layerwise latencies.

## 6.1 Pruning on Different Abstraction Levels

In this experiment, we fix the target hardware architecture to an Eyeriss-like 256 PE accelerator and perform pruning based on HW-Flow’s *Coarse*, *Mid* and *Fine* estimates. At each level, we choose the *constrained* reward function (equation 3), and set the target metric reduction to 50% of the baseline value (unpruned model execution). To observe the impact of the metrics at each abstraction level on the hardware architecture, we evaluate all the generated pruned networks on the *Fine*-level model.

The experiments in Table 5 serve as an example on how the three HW abstraction levels (*Course* | *Mid* | *Fine*) can be used to narrow down the range of CNNs and HW architectures which result in an optimal task-to-resource mapping. When using the HW-Flow design methodology, the *Coarse*-level helps the designer evaluate the pruning potential of a set of different CNNs. The designer only needs to have a rough number of OPs in mind, a target CNN memory footprint, and an estimation of the desired accuracy. For the purpose of demonstration, we use ResNet56 as our baseline CNN model, with a task-accuracy of 93.59% on the CIFAR-10 dataset. We analyze the compression capability by constraining 50% of OPs. After evaluating the CNNs’ compression potential, a set of promising candidates can be narrowed down.

The search can be refined to take the on-chip memory hierarchy and dimensioning into consideration at the *Mid*-level. The focus at this level is to choose the correct on-chip memory size and the amount of communication that needs to take place between the host and the accelerator. An under-dimensioned on-chip SRAM would lead to more stress on the communication infrastructure since more rounds of communication are necessary with the DRAM. A large SRAM, although costly, might relieve the complexity of a high-speed, high-bandwidth interconnect. In this context, HW-Flow’s *Mid*-level can play a pivotal role in helping the designer dimension the SRAM and the off-chip to on-chip interconnect while considering the pruning potential of the CNN. In Table 5, we check if the on-chip SRAM (256KB) is in a good range to achieve reductions in DRAM accesses without having to over-prune our CNN and lose the task-related accuracy goal with the available loop tiling and reordering possibilities. We observe that the agent prunes 63.84% of OPs constraining DRAM accesses to 50%.

Going deeper to the *Fine*-level, the pruning rate is relaxed as the efficient Eyeriss-like architecture is able to meet the constraint requirements without a high pruning rate. Consequently, this preserves the network’s accuracy equivalent to the *Coarse* level, while meeting lower target energy and latency.

■ **Table 5** ResNet56 pruned at different HW abstraction levels for Eyeriss-like 256 PE configurations. RS refers to row-stationary dataflow.

Prune configuration ( <code>&lt; constraint &gt;; &lt; level &gt;; &lt; hw_model &gt;</code> )	Acc [%]	PR [%]	Energy [ $\times 10^9$ ]	Latency [ $\times 10^3$ cycles]
Baseline (not pruned); 256 PE - RS	93.59	-	3.76	2350
-50% Ops *; Coarse; 256 PE - RS	93.03	50.00	2.08	1219
-50% DRAM access *; Mid; 256 PE - RS	91.82	63.84	1.50	862
-50% Energy *; Fine; 256 PE - RS	93.14	54.00	1.88	1159
-50% Latency *; Fine; 256 PE - RS	93.24	50.89	2.05	1176

\* : reduction required to meet constraint | (matched for)

## 6.2 Pruning on Different Rewards

To demonstrate the application of HW-Flow to a hardware design problem, we consider the three candidate Eyeriss-like hardware accelerators with 168, 256, and 1024 PEs. In this experiment, the agent performs pruning based on the two types of reward functions proposed in equation 3, namely estimate *constrained* and *balanced*.

**Estimate Constrained.** The agent is tasked with pruning the ResNet56 model trained on CIFAR-10 such that it meets a given fixed constraint while minimizing the accuracy degradation of the compressed network. The constraint is set to 50% energy or latency reduction relative to the baseline leader, i.e. the accelerator which performs the best for the target metric. The results in Table 6 show several interesting trends. We observe that the 168 PE variant is the baseline leader for energy-constrained pruning and the 1024 PE accelerator as a baseline leader for latency constrained pruning. With 1024 PEs, there is an ample capacity to improve latency, requiring a lower pruning rate to meet the application constraint. Conversely, the CNN can be pruned more effectively for 168 and 256 PEs when considering an energy-constrained application. For both cases, choosing the correct hardware platform results in a pruned network with higher accuracy. Figure 9a-c shows the agent’s decisions across the episodes for the three HW platforms with energy and latency constrained experiments. The noisy actions taken in the exploration phase (first 150 episodes) allow the agent to collect data on the environment and then start convergence and optimization in the following 400 episodes. For Figure 9a-latency and 9b-latency, the agent heavily prunes the model to achieve the target constraints, resulting in an accuracy degradation (marked as red in Table 6 if  $\geq 2\%$ ). In Figure 9c-energy, the agent struggles to meet the desired constraints, resulting in an accuracy degradation after fine-tuning. These critical observations can facilitate the choice of a suitable hardware for a given application constraint.

**Estimate Balanced.** As detailed in Section 4.2 and equation 3, the balanced estimate reward encourages the agent to maintain the target accuracy  $\psi^*$ , while minimizing the estimates  $\varphi$ . Here,  $\psi^*$  and  $b$  are set to 0.5, 0.125 respectively. Figure 10 demonstrates episode-wise reward plots for the balanced reward formulation of ResNet20 and ResNet56 configurations under various HW platforms. From Table 7, we observe that all the configurations optimized for energy and latency undergo minimal degradation in prediction accuracy with different latency and energy estimates. The Eyeriss-like 168 PE configuration achieves the best energy, whereas 1024 PEs achieves the best latency. Generally, for experiments in Figure 10, we observe an improvement in accuracy and reduction in HW metrics as the number of episodes increase. We also observe that the agent



■ **Table 6** Pruning ResNet56 on CIFAR-10 using estimate *constrained* reward  $\mathcal{R}$  on Eyeriss-like accelerators.

Prune configuration ( <code>&lt; constraint &gt;</code> ; <code>&lt; level &gt;</code> ; <code>&lt; hw_model &gt;</code> )	Acc [%]	PR [%]	Energy [ $\times 10^9$ ]	Latency [ $\times 10^3$ cycles]
Baseline (not pruned); Fine; 168 PE - RS	93.59	-	3.72	3377
Baseline (not pruned); Fine; 256 PE - RS	93.59	-	3.76	2350
Baseline (not pruned); Fine; 1024 PE - RS	93.59	-	5.52	588
Target Energy (-50%)**; Fine; 168 PE - RS*	92.63	58.16	1.85	1644
Target Energy (-50%)**; Fine; 256 PE - RS	93.14	54.00	1.88	1159
Target Energy (-66%)**; Fine; 1024 PE - RS	91.09	75.22	1.88	170
Target Latency (-92%)**; Fine; 168 PE - RS	86.89	93.14	0.40	269
Target Latency (-87%)**; Fine; 256 PE - RS	89.66	87.94	0.59	306
Target Latency (-50%)**; Fine; 1024 PE - RS*	92.92	52.68	3.07	294

\*: Baseline leader | \*\*: reduction required to meet constraint | (violated constraint) (matched constraint)

■ **Table 7** Pruning ResNet56 on CIFAR-10 using the estimate balanced reward function on Eyeriss-like accelerators.

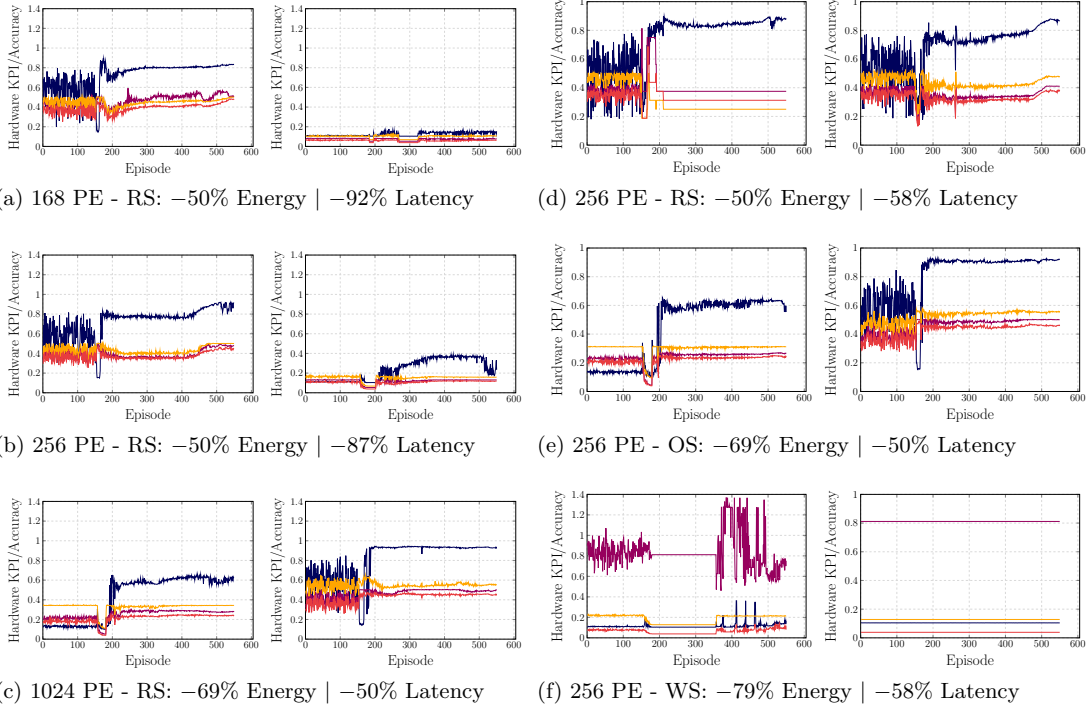
Prune configuration ( <code>&lt; reward &gt;</code> ; <code>&lt; level &gt;</code> ; <code>&lt; hw_model &gt;</code> )	Acc [%]	PR [%]	Energy [ $\times 10^9$ ]	Latency [ $\times 10^3$ cycles]
Baseline (not pruned); Fine; 168 PE - RS	93.59	-	3.72	3377
Baseline (not pruned); Fine; 256 PE - RS	93.59	-	3.76	2350
Baseline (not pruned); Fine; 1024 PE - RS	93.59	-	5.52	588
Energy balanced; Fine; 168 PE - RS	91.94	69.14	1.41	1309
Energy balanced; Fine; 256 PE - RS	92.56	62.22	1.61	913
Energy balanced; Fine; 1024 PE - RS	92.30	62.09	2.69	238
Latency balanced; Fine; 168 PE - RS	92.64	56.50	1.94	1658
Latency balanced; Fine; 256 PE - RS	92.58	59.75	1.69	975
Latency balanced; Fine; 1024 PE - RS	92.97	57.14	3.18	276

finds a pruning strategy for challenging HW configurations (168 PE latency constraint or 1024 PE energy constraint), with minimal accuracy degradation. We also observe quick convergence for ResNet56 pruning on the 256 PE accelerator in Figure 10e.

### 6.3 Pruning on Different Mappings

The following experiment is performed to evaluate the relationship between effective pruning and an efficient dataflow. We compare the target hardware model, with 256 PEs, against two variants with identical specification, except for their dataflows. Here, the three dataflows, weight-stationary (WS), output-stationary (OS), and row-stationary (RS), described in Section 4.4, are compared in their potential for improved execution of pruned CNNs.

The baseline estimates of the unpruned network show the energy and latency variation caused by dataflows (Table 8). All three dataflows present unique non-dominated solutions for baseline energy and latency. Similar to the estimate constrained experiment in Section 6.2, we set the constraint with respect to the baseline leader dataflow. Figure 9d-f shows the agent’s highly varying actions depending on the dataflow and the target constraint. RS results in the lowest baseline energy, whereas the OS has the lowest baseline latency. We can observe that the agent obtains minimum accuracy degradation for RS when constraining for energy. When constraining



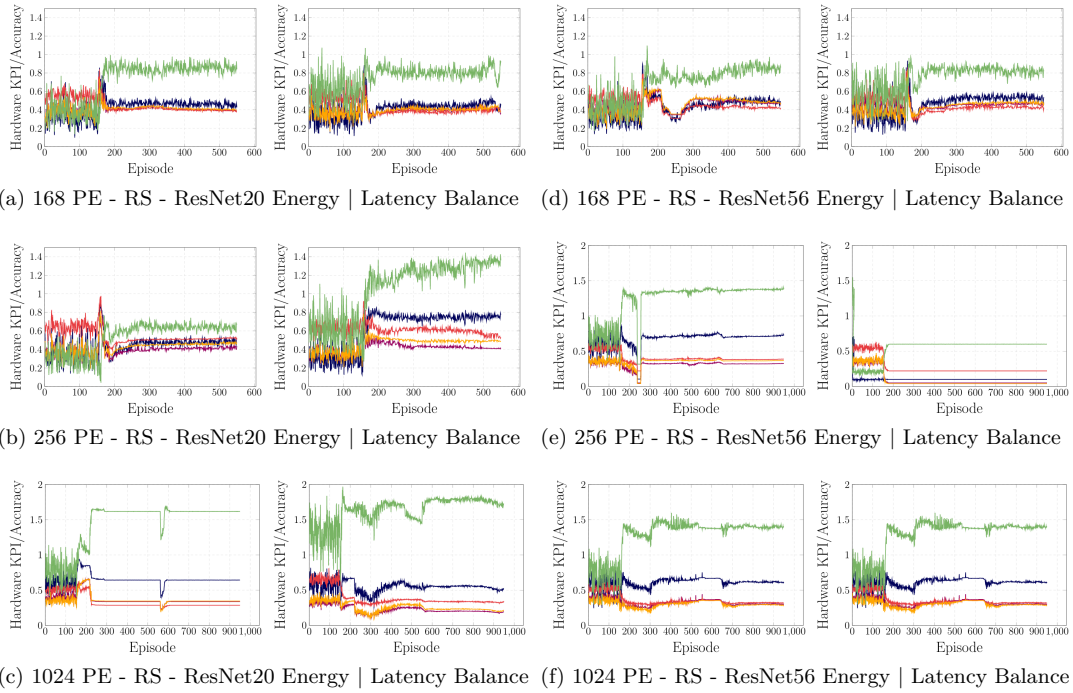
■ **Figure 9** Training curves of the agent detailing the █ Reward, reduction in █ Latency █ Energy █ OPs at every episode. We calculate reward by computing prediction accuracy on 10000 randomly sampled images from training dataset.

■ **Table 8** Constraining dataflows relative to 50% of the baseline leader (RS for energy and OS for latency).

Prune configuration ( <code>&lt; constraint &gt;; &lt; level &gt;; &lt; hw_model &gt;</code> )	Acc [%]	PR [%]	Energy [ $\times 10^9$ ]	Latency [ $\times 10^3$ cycles]
Baseline (not pruned); Fine; 256 PE - OS	93.59	-	5.87	1960
Baseline (not pruned); Fine; 256 PE - WS	93.59	-	5.77	1991
Baseline (not pruned); Fine; 256 PE - RS	93.59	-	3.76	2350
Target Energy (-68%); Fine; 256 PE - OS	91.84	72.05	1.88	584
Target Energy (-68%); Fine; 256 PE - WS	90.06	84.53	1.75	1308
Target Energy (-50%); Fine; 256 PE - RS *	93.14	54.00	1.88	1159
Target Latency (-50%); Fine; 256 PE - OS *	92.91	52.11	3.06	981
Target Latency (-51%); Fine; 256 PE - WS	84.17	96.20	0.71	1612
Target Latency (-58%); Fine; 256 PE - RS	92.36	61.05	1.72	984

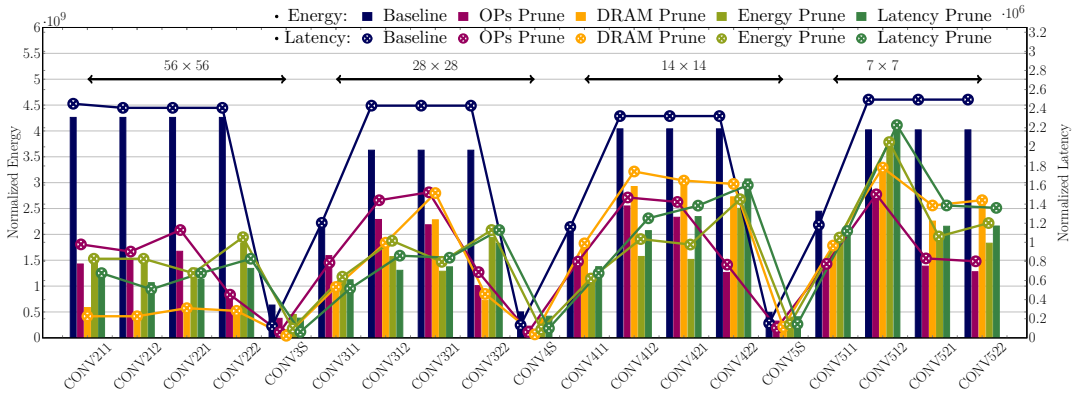
\*: Baseline leader | (violated constraint) (matched constraint)

for latency, the agent achieves better accuracy for OS and RS. WS demands higher pruning rate when constraining both energy and latency thereby resulting in lower accuracy (marked as red in Table 8). We can also see that the agent does not change its action across several episodes when constraining latency under WS dataflow (see Figure 9f). Thus, we can conclude that the row stationary dataflow is an optimal mapping scheme to achieve efficient energy and latency.



■ **Figure 10** Training curves of the agent detailing the — Reward — Accuracy — Normalized Latency — Normalized Energy — Normalized OPs at every episode.

### 6.4 Layer-wise Analysis of ResNet18



■ **Figure 11** Energy consumption and latency of the pruned layers in ResNet18 on an Eyeriss-like 256 PE accelerator under different pruning constraints.

Based on different pruning constraints, the layer-wise analysis of the ImageNet-trained ResNet18, scheduled on an Eyeriss-like 256 PE accelerator, is presented in Figure 11. The architecture of ResNet18 consists of four stages based on the output feature map spatial size. The results detail the achieved *Fine* estimates (normalized energy and latency). We observe that the agent’s pruning rate decision for each layer depends on the constrained HW metric. The layers with higher spatial output sizes ( $56 \times 56$ ) are aggressively pruned when constraining DRAM

accesses. On the other hand, the pruning rate for the layers with smaller spatial output size ( $7 \times 7$ ) is observed higher when constraining for OPs. We also observe a lower drop in energy and latency ( $\leq 50\%$ ) for layers such as CONV411, CONV511, CONV512 to avoid accuracy degradation for all kinds of pruning constraints. The prediction accuracy and the pruning rates of the four configurations are reported in Table 10.

## 6.5 Pruning DeepLabv3 for Semantic Segmentation

Using the HW-Flow estimations, we prune DeepLabv3 [3] (using ResNet18 backbone) on the CityScapes dataset. For the DeepLab-based CNN, the bottleneck layers consist of two residual blocks with a dilation rate of 2 and an Atrous Spatial Pyramid Pooling (ASPP) block with dilation rates  $\{1, 8, 12, 18\}$ . To obtain *Fine*-level estimates for dilated convolutional layers from the HW-Flow framework, we adapt the row-stationary dataflow. The rows of PEs responsible for the dilated parts of the kernel can either be clock-gated or removed from the logical mapping. This implies that the diagonal reuse of input pixels across the spatial array is disrupted. This phenomenon is equivalent to a regular convolution with a large stride, where not every row of the input feature is shared directly with the diagonal neighbor PE [4]. Nevertheless, a non-direct neighbor PE may still reuse the input feature map row. In this case, the potential to reuse an input feature map row at the PE array-level depends on the degree of unrolling  $P_{Ho}$ , the dilation rate, and the stride. We use an Eyeriss-like architecture with a large PE array to perform inference of the DeepLabv3 model. In Table 9, we highlight that the DeepLabv3 cannot be scheduled on the standard Eyeriss architecture [4] (Eyeriss-168). This is due to the *ifmap* register files being dimensioned to hold at-most 12 pixels at a time (see Table 3), which is a decision made by the designers in [4] to support the largest kernel size row in AlexNet (11 pixels). The dilated convolution layers in DeepLabv3, can have up to 36 pixel rows at a time, for a  $3 \times 3$  kernel with a dilation rate of 18. Increasing the PE array dimensions would not resolve this issue, as it is inherent to the pipeline and dataflow constraints of the Eyeriss-like architecture. We increase the *ifmap* register sizes to 37 pixels per PE (i.e.  $36 + 1$ ) to make all layers schedulable on the accelerator and obtain baseline estimates.

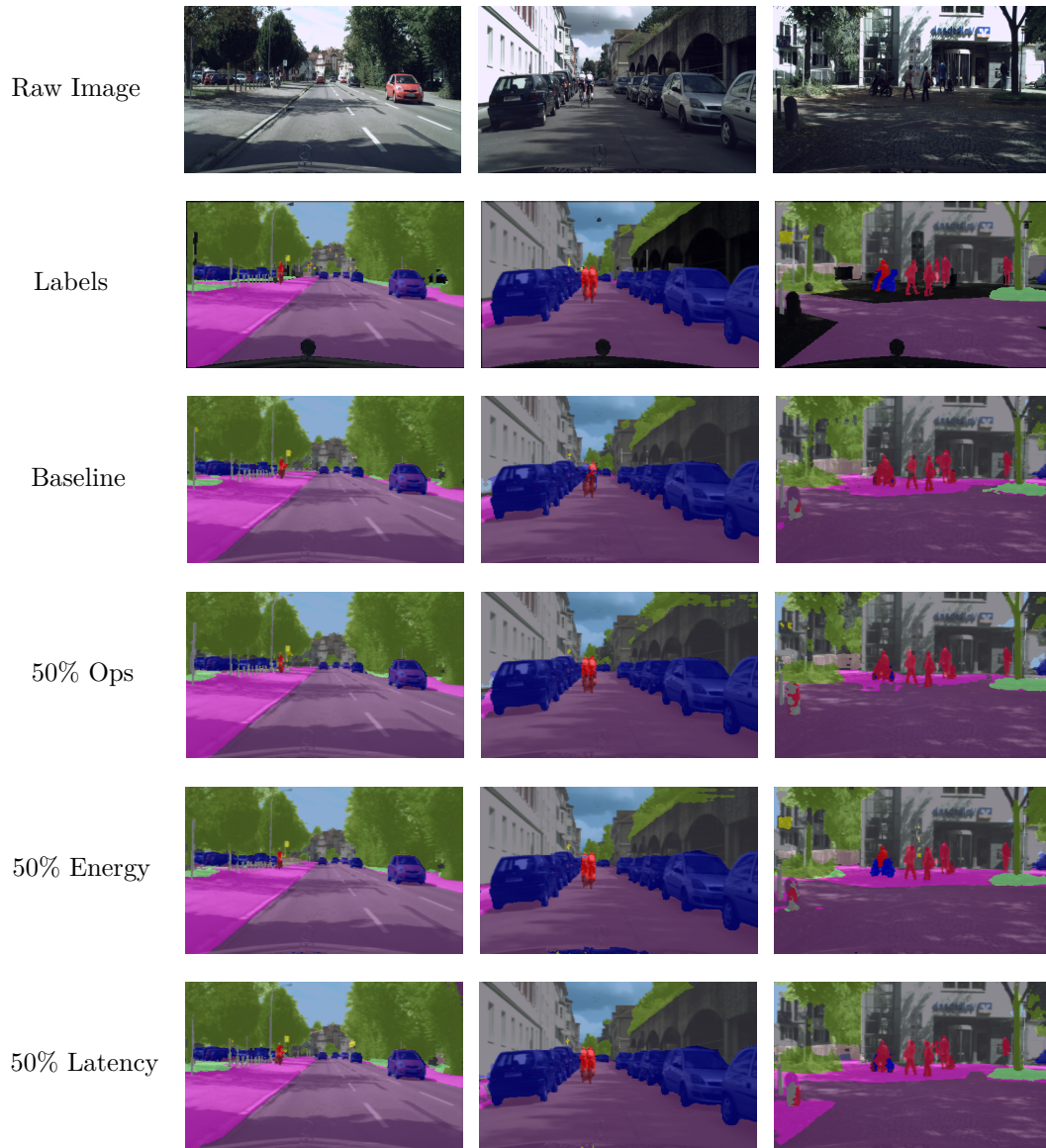
■ **Table 9** Pruning DeepLabv3 on the CityScapes dataset.

Prune configuration ( <code>&lt; reward &gt;</code> ; <code>&lt; level &gt;</code> ; <code>&lt; hw_model &gt;</code> )	mIOU [%]	PR [%]	Memory [MB]	Energy [ $\times 10^9$ ]	Latency [ $\times 10^6$ cycles]
Baseline (not pruned); Fine; Eyeriss-like 168 PE	69.68	-	33.26	NS	NS
Baseline (not pruned); Fine; Eyeriss-like 1024 PE	69.68	-	33.26	NS	NS
Baseline (not pruned); Fine; Eyeriss-like-Deeplab	69.68	-	33.26	1541	267.4
Ops Constrained (Ours); Coarse; Eyeriss-like-Deeplab	69.69	50.00	25.48	954	174.9
Energy Constrained (Ours); Fine; Eyeriss-like-Deeplab	69.88	51.90	29.05	820	161.5
Latency Constrained(Ours); Fine; Eyeriss-like-Deeplab	69.79	60.36	16.87	677	119.6

(NS: Not Schedulable) (matched constraint)

We constrain the number of operations, energy, and latency during the pruning process to 50% as shown in Table 9. There is no degradation in the mIOU (mean intersection over union) evaluation metric for different pruning constraints. We could derive that the unpruned DeepLabv3 model is over-parameterized for the CityScapes dataset. We observe that a higher pruning rate is required to constrain latency to 50%. We highlight the pruned models' effectiveness by demonstrating the semantic predictions on three sample images of the CityScapes dataset. We observe that the pruned models could produce better predictions (terrain in column 1, bikers in column 2, motorcycle and terrain in column 3) due to their higher generalization capability. By analyzing the layer-wise pruning ratios for different target constraints, we observe that the agent heavily prunes the ASPP and decoder blocks. For energy-constrained pruning, the agent only finds redundant operations in the decoder blocks.

### 03:26 HW-Flow: A Multi-Abstraction Level HW-CNN Codesign Pruning Methodology



■ **Figure 12** Qualitative results for pruned models on different scenarios in the CityScapes dataset. Black regions are unlabeled in the original dataset.



## 6.6 Results Summary and Discussion

In this section, we compare HW-Flow with other channel pruning works proposed in literature. Table 10 details various pruning configurations of ResNet variants trained on CIFAR-10 (ResNet20, ResNet56) and ImageNet (ResNet18, ResNet50), evaluated on an Eyeriss-like 256 PE accelerator. The table also includes other pruning methods and baseline models for reference. FPGM [15] and Channel-Pruning [14] do not consider HW metrics as optimization targets or constraints, but rather limit the compressed models only to be efficient with respect to computational complexity. Due to a lack of information given by the authors, the estimation of the energy and the latency is not possible for these works.

■ **Table 10** Constrained and balanced pruning configurations using HW-Flow on ResNet variants, compared to other works in literature. HW estimates measured on Eyeriss-like-256.

Prune configuration ( $\langle \text{reward} \rangle; \langle \text{level} \rangle$ )	Acc [%]	PR [%]	Memory [MB]	Energy [ $\times 10^9$ ]	Latency [ $\times 10^3$ cycles]
<b>ResNet20</b>					
Baseline (not pruned)	92.48	-	0.54	1.22	765
FPGM [15] (HW agnostic)	91.09	42.20	-	-	-
Ops Constrained [16]; Coarse	91.78	50.00	0.33	0.82	464
DRAM Constrained (Ours); Mid	90.78	70.87	0.27	0.43	236
Energy Constrained (Ours); Fine	91.46	56.12	0.35	0.61	359
Latency Constrained (Ours); Fine	90.53	48.55	0.35	0.66	383
<b>ResNet56</b>					
Baseline (not pruned)	93.59	-	1.69	3.76	2350
FPGM [15] (HW agnostic)	92.89	52.60	-	-	-
Channel-pruning [14] (proxy)	91.80	50.00	-	-	-
Ops Constrained [16]; Coarse	93.03	50.00	1.21	2.08	1219
DRAM Constrained (Ours); Mid	91.82	63.84	1.21	1.50	862
Energy Constrained (Ours); Fine	93.14	54.00	1.11	1.88	1159
Latency Constrained (Ours); Fine	93.24	50.89	1.15	2.05	1176
<b>ResNet18</b>					
Baseline (not pruned)	68.33	-	23.34	64.85	37796
FPGM [15] (HW agnostic)	67.81	41.80	-	-	-
Ops Constrained [16]; Coarse	67.66	50.00	16.94	32.89	18906
DRAM Constrained (Ours); Mid	66.38	54.46	15.94	30.17	16755
Energy Constrained (Ours); Fine	66.58	50.63	14.69	32.32	18280
Latency Constrained(Ours); Fine	66.92	49.70	16.61	33.62	18889
<b>ResNet50</b>					
Baseline (not pruned)	76.06	-	51.00	361.12	206873
FPGM [15]	74.83	53.50	-	-	-
Channel-pruning [14] (proxy)	72.30	50.00	-	-	-
Ops Constrained [16]; Coarse	73.25	50.00	22.67	178.55	103968
DRAM Constrained (Ours); Mid	72.17	58.61	17.16	148.67	86411
Energy Constrained (Ours); Fine	73.69	49.82	24.12	180.91	104411
Latency Constrained(Ours); Fine	74.35	49.68	25.62	180.93	103576

(violated constraint) (matched constraint)

The accuracy and HW-estimates for AMC [16] are re-implemented by constraining the OPs in HW-Flow’s *Coarse*-level estimation. We observe that constraining DRAM accesses by 50% using *Mid*-level estimation demands higher pruning rate as there is little room for optimizing the HW schedule. This results in accuracy degradation compared to other pruning configurations from other target constraints (see DRAM constrained pruning for ResNet56, ResNet50 in Table 10). HW-Flow is able to constrain energy and latency precisely to 50% of its baseline metrics by using *Fine*-level HW estimates during the pruning process. For ResNet50, the energy and latency

constrained solutions by HW-Flow produce 0.44% and 1.10% better prediction accuracy compared to the work in AMC (OPs constrained). AMC also prunes the CNNs based on latency, but it is only limited to general-purpose HW platforms (Pixel 1 and TitanX GPU). We should note that HW-Flow can prune CNN models at different HW abstraction levels and with a customizable, accurate HW optimizer/modeler, thus allowing for a HW/CNN co-design approach.

## 7 Conclusion

Optimization of CNNs and the design of resource-constrained HW platforms go hand in hand. In this paper, we propose HW-Flow, a framework for optimizing and exploring CNN models based on three levels of hardware abstraction: *Coarse*, *Mid* and *Fine*. We propose analytical search techniques to systematically traverse through the scheduling and mapping space, thereby generating accurate HW estimates. We show that the pruning rate is an inaccurate proxy metric for HW efficiency. With HW-aware pruning using *Fine*-level estimates, HW-Flow achieved  $\times 2$  energy and latency reduction with minimal loss in prediction accuracy compared to its baseline unpruned models. We extend the investigation to segmentation tasks, where observations on pruning rates of decoder and ASPP blocks were made with respect to the pruning target. DeepLabv3's energy and latency were reduced by  $\sim 50\%$ , while improving the accuracy of the baseline, over-parameterized model. HW-Flow can prune CNN models at different HW abstraction levels and with a customizable and accurate HW modeling technique, facilitating a HW-CNN codesign approach.

## References

- 1 Han Cai, Ligeng Zhu, and Song Han. ProxylessNAS: Direct Neural Architecture Search on Target Task and Hardware. In *International Conference on Learning Representations (ICLR)*, 2019. URL: <https://dblp.org/rec/conf/iclr/CaiZH19.bib>.
- 2 C. Chen, A. Seff, A. Kornhauser, and J. Xiao. Deep-driving: Learning affordance for direct perception in autonomous driving. In *2015 IEEE International Conference on Computer Vision (ICCV)*, pages 2722–2730, December 2015. doi:10.1109/ICCV.2015.312.
- 3 Liang-Chieh Chen, Yukun Zhu, George Papandreou, Florian Schroff, and Hartwig Adam. Encoder-Decoder with Atrous Separable Convolution for Semantic Image Segmentation. In *The European Conference on Computer Vision (ECCV)*, Cham, 2018. Springer International Publishing. doi:10.1007/978-3-030-01234-2\_49.
- 4 Y. Chen, J. Emer, and V. Sze. Eyeriss: A Spatial Architecture for Energy-Efficient Dataflow for Convolutional Neural Networks. In *ACM/IEEE Annual International Symposium on Computer Architecture (ISCA)*, 2016. doi:10.1109/ISCA.2016.40.
- 5 Marius Cordts, Mohamed Omran, Sebastian Ramos, Timo Rehfeld, Markus Enzweiler, Rodrigo Benenson, Uwe Franke, Stefan Roth, and Bernt Schiele. The Cityscapes Dataset for Semantic Urban Scene Understanding. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016. URL: <https://dblp.org/rec/journals/corr/CordtsORREBF16.bib>.
- 6 Yann Le Cun, John S. Denker, and Sara A. Solla. Optimal Brain Damage. In *Advances in Neural Information Processing Systems (NeurIPS)*. Morgan Kaufmann Publishers Inc., 1990.
- 7 Xiaoliang Dai, Peizhao Zhang, Bichen Wu, Hongxu Yin, Fei Sun, Yanghan Wang, Marat Dukhan, Yunqing Hu, Yiming Wu, Yangqing Jia, Peter Vajda, Matthew Uyttendaele, and Niraj K. Jha. Chamnet: Towards efficient network design through platform-aware model adaptation. *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2019. doi:10.1109/CVPR.2019.01166.
- 8 Xuanyi Dong and Yi Yang. Nas-bench-201: Extending the scope of reproducible neural architecture search. In *International Conference on Learning Representations*, 2020. URL: <https://openreview.net/forum?id=HJxyZkBKDr>.
- 9 Alexander Frickenstein, Manoj-Rohit Vemparala, Nael Fafous, Laura Hauenschild, Naveen-Shankar Nagaraja, Christian Unger, and Walter Stechele. Alf: Autoencoder-based low-rank filter-sharing for efficient convolutional neural networks. In *Proceedings of the 57th ACM/EDAC/IEEE Design Automation Conference, (DAC)*, 2020. doi:10.1109/DAC18072.2020.9218501.
- 10 Yiwen Guo, Anbang Yao, and Yurong Chen. Dynamic Network Surgery for Efficient DNNs. In D. D. Lee, M. Sugiyama, U. V. Luxburg, I. Guyon, and R. Garnett, editors, *Advances in Neural Information Processing Systems (NeurIPS)*. Curran Associates, Inc., 2016. URL: <https://dblp.org/rec/conf/nips/GuoYC16.bib>.
- 11 Song Han, Jeff Pool, John Tran, and William Dally. Learning both Weights and Connections for Efficient Neural Network. In C. Cortes, N. D. Lawrence, D. D. Lee, M. Sugiyama, and R. Garnett, editors,



- Advances in Neural Information Processing Systems (NeurIPS)*. Curran Associates, Inc., 2015.
- 12 Babak Hassibi, David G. Stork, Gregory Wolff, and Takahiro Watanabe. Optimal Brain Surgeon: Extensions and Performance Comparisons. In *Advances in Neural Information Processing Systems (NeurIPS)*, San Francisco, CA, USA, 1993. Morgan Kaufmann Publishers Inc. doi:10.1109/ICNN.1993.298572.
  - 13 K. He, X. Zhang, S. Ren, and J. Sun. Deep Residual Learning for Image Recognition. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016. doi:10.1109/CVPR.2016.90.
  - 14 Y. He, X. Zhang, and J. Sun. Channel Pruning for Accelerating Very Deep Neural Networks. In *IEEE International Conference on Computer Vision (ICCV)*, 2017. doi:10.1109/ICCV.2017.155.
  - 15 Yang He, Ping Liu, Ziwei Wang, Zhilan Hu, and Yang Yang. Filter Pruning via Geometric Median for Deep Convolutional Neural Networks Acceleration. *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2018. doi:10.1109/CVPR.2019.00447.
  - 16 Yihui He, Ji Lin, Zhijian Liu, Hanrui Wang, Li-Jia Li, and Song Han. AMC: AutoML for Model Compression and Acceleration on Mobile Devices. In *European Conference on Computer Vision (ECCV)*, 2018. doi:10.1007/978-3-030-01234-2\_48.
  - 17 Qianguo Huang, Shaohua Kevin Zhou, Suyu You, and Ulrich Neumann. Learning to Prune Filters in Convolutional Neural Networks. *IEEE Winter Conference on Applications of Computer Vision (WACV)*, 2018. doi:10.1109/WACV.2018.00083.
  - 18 Leslie Pack Kaelbling, Michael L Littman, and Andrew W Moore. Reinforcement learning: A survey. *Journal of artificial intelligence research*, 4:237–285, 1996.
  - 19 Alex Krizhevsky. Learning Multiple Layers of Features from Tiny Images, 2009. University of Toronto.
  - 20 Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. In F. Pereira, C. J. C. Burges, L. Bottou, and K. Q. Weinberger, editors, *Advances in Neural Information Processing Systems*, volume 25. Curran Associates, Inc., 2012. URL: <https://proceedings.neurips.cc/paper/2012/file/c399862d3b9d6b76c8436e924a68c45b-Paper.pdf>.
  - 21 Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. ImageNet Classification with Deep Convolutional Neural Networks. In F. Pereira, C. J. C. Burges, L. Bottou, and K. Q. Weinberger, editors, *Advances in Neural Information Processing Systems (NeurIPS)*. Curran Associates, Inc., 2012. doi:10.1145/3065386.
  - 22 Chaojian Li, Zhongzhi Yu, Yonggan Fu, Yonggan Zhang, Yang Zhao, Haoran You, Qixuan Yu, Yue Wang, Cong Hao, and Yingyan Lin. {HW}-{nas}-bench: Hardware-aware neural architecture search benchmark. In *International Conference on Learning Representations*, 2021. URL: [https://openreview.net/forum?id=\\_0kaDkv3dVf](https://openreview.net/forum?id=_0kaDkv3dVf).
  - 23 Timothy P Lillicrap, Jonathan J Hunt, Alexander Pritzel, Nicolas Heess, Tom Erez, Yuval Tassa, David Silver, and Daan Wierstra. Continuous control with deep reinforcement learning. *arXiv preprint arXiv:1509.02971*, 2015.
  - 24 Shuying Liu and Weihong Deng. Very deep convolutional neural network based image classification using small training sample size. In *2015 3rd IAPR Asian Conference on Pattern Recognition (ACPR)*, pages 730–734, 2015. doi:10.1109/ACPR.2015.7486599.
  - 25 Jonathan Long, Evan Shelhamer, and Trevor Darrell. Fully convolutional networks for semantic segmentation. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2015.
  - 26 Hongzi Mao, Mohammad Alizadeh, Ishai Menache, and Srikanth Kandula. Resource management with deep reinforcement learning. In *Proceedings of the 15th ACM workshop on hot topics in networks*, pages 50–56, 2016.
  - 27 A. Parashar, P. Raina, Y. S. Shao, Y. Chen, V. A. Ying, A. Mukkara, R. Venkatesan, B. Khailany, S. W. Keckler, and J. Emer. Timeloop: A Systematic Approach to DNN Accelerator Evaluation. In *IEEE International Symposium on Performance Analysis of Systems and Software (ISPASS)*, 2019. doi:10.1109/ISPASS.2019.00042.
  - 28 S. Pereira, A. Pinto, V. Alves, and C. A. Silva. Brain tumor segmentation using convolutional neural networks in mri images. *IEEE Transactions on Medical Imaging*, 35(5):1240–1251, May 2016. doi:10.1109/TMI.2016.2538465.
  - 29 Martin Riedmiller and Thomas Gabel. On experiences in a complex and competitive gaming domain: Reinforcement learning meets robocup. In *2007 IEEE Symposium on Computational Intelligence and Games*, pages 17–23, 2007.
  - 30 Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael Bernstein, Alexander C. Berg, and Li Fei-Fei. ImageNet Large Scale Visual Recognition Challenge. *International Journal of Computer Vision (IJCV)*, 115(3), 2015. doi:10.1007/s11263-015-0816-y.
  - 31 Hardik Sharma, Jongse Park, Naveen Suda, Liangzhen Lai, Benson Chau, Vikas Chandra, and Hadi Esmailzadeh. Bit Fusion: Bit-Level Dynamically Composable Architecture for Accelerating Deep Neural Networks. In *ACM/IEEE Annual International Symposium on Computer Architecture (ISCA)*, ISCA '18. IEEE Press, 2018. doi:10.1109/ISCA.2018.00069.
  - 32 V. Sze, Y. Chen, T. Yang, and J. S. Emer. Efficient Processing of Deep Neural Networks: A Tutorial and Survey. *Proceedings of the IEEE (Volume: 105, Issue: 12)*, 105(12), November 2017.
  - 33 Christian Szegedy, W. Liu, Y. Jia, Pierre Sermanet, Scott E. Reed, Dragomir Anguelov, D. Erhan, V. Vanhoucke, and Andrew Rabinovich. Going deeper with convolutions. *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 1–9, 2015.
  - 34 Mingxing Tan and Quoc Le. EfficientNet: Rethinking model scaling for convolutional neural networks. In *Proceedings of the 36th International Conference on Machine Learning*, 2019.

- 35 F. Tu, S. Yin, P. Ouyang, S. Tang, L. Liu, and S. Wei. Deep convolutional neural network architecture with reconfigurable computation patterns. *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, 25(8):2220–2233, 2017. doi:10.1109/TVLSI.2017.2688340.
- 36 Manoj Rohit Vemparala, Nael Fasfous, Alexander Frickenstein, Sreetama Sarkar, Qi Zhao, Sabine Kuhn, Lukas Frickenstein, Anmol Singh, Christian Unger, Naveen Shankar Nagaraja, Christian Wressnegger, and Walter Stechele. Adversarial robust model compression using in-train pruning. *2021 IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*, pages 66–75, 2021.
- 37 R. Venkatesan, Y. Shao, Miaorong Wang, Jason Clemons, S. Dai, M. Fojtik, Ben Keller, Alicia Klinefelter, N. Pinckney, Priyanka Raina, Y. Zhang, B. Zimmer, W. Dally, J. Emer, Stephen W. Keckler, and B. Khailany. Magnet: A modular accelerator generator for neural networks. In *2019 IEEE/ACM International Conference on Computer-Aided Design (ICCAD)*, 2019. doi:10.1109/ICCAD45719.2019.8942127.
- 38 Kuan Wang, Zhijian Liu, Yujun Lin, Ji Lin, and Song Han. HAQ: Hardware-Aware Automated Quantization With Mixed Precision. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2019.
- 39 Marco A Wiering. Multi-agent reinforcement learning for traffic light control. In *Machine Learning: Proceedings of the Seventeenth International Conference (ICML'2000)*, pages 1151–1158, 2000.
- 40 Bichen Wu, Xiaoliang Dai, Peizhao Zhang, Yanghan Wang, Fei Sun, Yiming Wu, Yuandong Tian, Péter Vajda, Yangqing Jia, and Kurt Keutzer. Fbnet: Hardware-aware efficient convnet design via differentiable neural architecture search. *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 10726–10734, 2019.
- 41 T. Yang, Y. Chen, and V. Sze. Designing Energy-Efficient Convolutional Neural Networks Using Energy-Aware Pruning. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017. doi:10.1109/CVPR.2017.643.
- 42 Tien-Ju Yang, Andrew Howard, Bo Chen, Xiao Zhang, Alec Go, Mark Sandler, and Hartwig Sze, Vivienne and Adam. NetAdapt: Platform-Aware Neural Network Adaptation for Mobile Applications. In *The European Conference on Computer Vision (ECCV)*. Springer International Publishing, 2018. URL: <https://dblp.org/rec/journals/corr/abs-1804-03230.bib>.
- 43 Xuan Yang, Mingyu Gao, Qiaoyi Liu, Jeff Setter, Jing Pu, Ankita Nayak, Steven Bell, Kaidi Cao, Heonjae Ha, Priyanka Raina, Christos Kozyrakis, and Mark Horowitz. Interstellar: Using halide’s scheduling language to analyze dnn accelerators. In *International Conference on Architectural Support for Programming Languages and Operating Systems (ASPLOS)*, pages 369–383, New York, NY, USA, 2020. Association for Computing Machinery. doi:10.1145/3373376.3378514.
- 44 C. Zhang, Zhenman Fang, Peipei Zhou, Peichen Pan, and Jason Cong. Caffeine: Towards uniformed representation and acceleration for deep convolutional neural networks. In *IEEE/ACM International Conference on Computer-Aided Design (ICCAD)*, 2016. doi:10.1145/2966986.2967011.
- 45 Xingyi Zhou, Dequan Wang, and Philipp Krähenbühl. Objects as points. In *arXiv preprint*, 2019. arXiv:1904.07850.