DISS. ETH NO. 29092

Designing Efficient Deep Neural Networks:
Topological Optimization, Quantization and Multi-Task Learning

A thesis submitted to attain the degree of

DOCTOR OF SCIENCES of ETH Zürich
(Dr. sc. ETH Zürich)

presented by

Menelaos Kanakis

Master of Science ETH in Robotics, Systems and Control
ETH Zürich
born 8 September 1991
citizen of Cyprus and the United Kingdom

accepted on the recommendation of

Prof. Dr. Luc Van Gool, examiner
Prof. Dr. Margarita Chli, co-examiner
Prof. Dr. Hakan Bilen, co-examiner
Dr. Ajad Chhatkuli, co-examiner

2023

To my family!

ACKNOWLEDGMENTS

First of all, I would like to express my deepest gratitude to Prof. Luc Van Gool for the opportunity to conduct my doctoral studies in the Computer Vision Lab at ETH Zurich under his supervision. He provided us with a wonderful environment to conduct research in, and personally with the freedom to work on the topics I was most motivated and interested in.

I would like to thank Prof. Margarita Chli and Prof. Hakan Bilen for promptly agreeing to be part of my examination committee, as well as their constructive feedback.

Furthermore, I would like to thank a large number of people that helped me, in different ways, become the researcher I am today. Specifically, and in no specific order,

— **My closest CVL/ETH collaborators for their invaluable contributions:** David Bruggemann, Ajad Chhatkuli, Anton Obukhov, Matteo Spallanzani, Martin Danelljan, Suman Saha, Stamatios Georgoulis, Thomas E. Huang, and Fisher Yu.

— **Additional CVL collaborators:** Danda Pani Paudel, Guolei Sun, Thomas Probst, Nikola Popovic, Yuhua Chen, and Dengxin Dai.

— **Further CVL colleagues and friends:** Evangelos Ntavelis, Despoina Paschalidou, Gustav Bredell, Christoph Mayer, Dario Fuoli, Goutam Bhat, Mohamad Shahbazi, Clara Fernandez-Labrador, Vasileios Choutas, Prune Truong, Alex Liniger, and Simon Hecker.

— **All my students for the very pleasant collaborations:** Philippe Blatter, Simon Maurer, Zador Pataki, Tianfu Wang, Konstantinos Skovolas, Qasim Warraich, Rohit Koush, and Wenhao Xu.

— **The earlier, and very talented, PhD students of our lab that inspired me early on in my PhD journey:** Kevis-Kokitsi Maninis, Christos Sakaridis, Sergi Caelles, Fabian Mentzer, and Thomas Probst, amongst many others.

— **The labs administration office for all their help:** Christina Krueger, Christine Braun-Roth, and Kristine Haberer.

Of course, none of this would be possible, nor as meaningful, without the unconditional love and support of my family, and especially my parents Kanakis and Rosina. The most special thanks goes to Nicole, who has made my life so enjoyable and for always being there.

ABSTRACT

The design of more complex and powerful deep neural networks has consistently advanced the state-of-the-art in a wide range of tasks over time. In the pursuit of increased performance, computational complexity is often severely hindered, as seen by the significant increase in the number of parameters, the required floating-point operations, and latency. While the great advancements of deep neural networks increase the interest in their use in downstream applications such as robotics and augmented reality, these applications require computationally efficient alternatives. This thesis focuses on the design of efficient deep neural networks, specifically, improving performance given computational constraints, or decreasing complexity with minor performance degradation.

Firstly, we present a novel convolutional operation reparameterization and its application to multi-task learning. By reparameterizing the convolutional operations, we can achieve comparable performance to single-task models at a fraction of the total number of parameters.

Secondly, we conduct an extensive study to evaluate the efficacy of self-supervised tasks as auxiliary tasks in a multi-task learning framework. We find that jointly training a target task with self-supervised tasks can improve performance and robustness, commonly outperforms labeled auxiliary tasks, while not requiring modifications to the architecture used at deployment.

Thirdly, we propose a novel transformer layer for efficient single-object visual tracking. We demonstrate that the performance of real-time single-object trackers can be significantly improved without compromising latency, while consistently outperforming alternative transformer layers.

Finally, we investigated the efficacy of adapting interest point detection and description neural networks for use in computationally limited platforms. We find that mixed-precision quantization of network components, coupled with a binary descriptor normalization layer, yields minor performance degradations while improving the size of sparse 3D maps, matching speed, and inference speed by at least an order of magnitude.

To conclude, this thesis focuses on the design of deep neural networks given computational limitations. With an increasing interest and demand for efficient deep networks, we envision the presented work will pave the way towards even more efficient methods, bridging the gap with better-performing alternatives.

ZUSAMMENFASSUNG

Die Entwicklung komplexerer und leistungsfähigerer tiefer neuronaler Netze hat den Stand der Technik bei einer Vielzahl von Aufgaben im Laufe der Zeit immer weiter verbessert. Jedoch wird das Streben nach höherer Leistung oft durch die Komplexität der Berechnungen stark behindert, was sich in einer erheblichen Zunahme der Anzahl der Parameter, der erforderlichen Gleitkommaoperationen und der Latenzzeit zeigt. Obwohl die grossen Fortschritte der tiefen neuronalen Netze das Interesse an ihrem Einsatz in Anwendungen wie Robotik und Augmented Reality erhöhen, erfordern diese Anwendungen recheneffiziente Alternativen. Diese Arbeit befasst sich mit dem Entwurf effizienter tiefer neuronaler Netze, insbesondere mit der Verbesserung der Leistung bei begrenztem Rechenaufwand oder mit der Verringerung der Komplexität bei geringerer Leistungseinbusse.

Zunächst stellen wir eine neuartige Umparametrisierung von Faltungsoperationen und ihre Anwendung für das Multi-Task-Lernen vor. Durch die Umparametrisierung der Faltungsoperationen können wir mit einem Bruchteil der Gesamtzahl der Parameter eine vergleichbare Leistung wie bei Single-Task-Modellen erzielen.

Zweitens führen wir eine umfassende Studie durch, um die Wirksamkeit von selbstüberwachten Aufgaben als Hilfslernsignale in einem Multi-Task-Lernsystem zu erfassen. Wir stellen fest, dass das gemeinsame Training einer Zielaufgabe mit selbstüberwachten Aufgaben die Leistung und Robustheit verbessern kann und in der Regel besser abschneidet als gelabelte Hilfsaufgaben, ohne dass Änderungen an der verwendeten Architektur erforderlich sind.

Drittens schlagen wir eine neuartige Transformatorschicht für das effiziente visuelle Tracking von Einzelobjekten vor. Wir zeigen, dass die Leistung von Echtzeit-Einzelobjekt-Trackern erheblich verbessert werden kann, ohne deren Latenz zu beeinträchtigen. Zugleich schneidet unsere Transformatorschicht durchgehend besser ab als alternative Transformatorschichten.

Schließlich untersuchen wir Anpassungen von neuronalen Erkennungs- und Beschreibungsnetzwerken für den Einsatz auf rechenschwachen Plattformen. Wir stellen fest, dass die Quantisierung der Netzkomponenten mit gemischter Genauigkeit in Verbindung mit einer binären Deskriptor-Normalisierungsschicht zu geringen Leistungseinbussen führt und gleich-

zeitig die Größe, Anpassungsgeschwindigkeit und Inferenzgeschwindigkeit des 3D-Modells um mindestens eine Grössenordnung verbessert.

Zusammengefasst formuliert, konzertriet sich diese Arbeit auf den Entwurf von tiefen neuronalen Netzen bei begrenzter Rechenleistung. Angesichts des zunehmenden Interesses und der Nachfrage nach effizienten tiefen Netzen gehen wir davon aus, dass die vorgestellte Arbeit den Weg zu noch effizienteren Methoden ebnen und die Lücke zu leistungsfähigeren Alternativen schließen wird.

PUBLICATIONS

The following publications are included in parts or in an extended version in this thesis:

— Menelaos Kanakis, David Bruggemann, Suman Saha, Stamatios Georgoulis, Anton Obukhov, and Luc Van Gool, "Reparameterizing Convolutions for Incremental Multi-Task Learning without Task Interference", ECCV, 2020 [Kan+20].

— Menelaos Kanakis, Thomas E. Huang, David Bruggemann, Fisher Yu, and Luc Van Gool, "Composite Learning for Robust and Effective Dense Predictions", WACV, 2023 [Kan+23].

— Philippe Blatter*, Menelaos Kanakis*, Martin Danelljan, and Luc Van Gool, "Efficient Visual Tracking with Exemplar Transformers", WACV, 2023 [Bla+23].

— Menelaos Kanakis*, Simon Maurer*, Matteo Spallanzani, Ajad Chhatkuli, and Luc Van Gool, "ZippyPoint: Fast Interest Point Detection, Description, and Matching through Mixed Precision Discretization", arXiv preprint arXiv:2203.03610, 2022 [Kan+22].

Furthermore, the following publications were co-authored during the period of my doctoral studies, however, are not covered in this thesis:

— Anton Obukhov, Maxim Rakhuba, Stamatios Georgoulis, Menelaos Kanakis, Dengxin Dai, and Luc Van Gool, "T-Basis: a Compact Representation for Neural Networks", ICML, 2020 [Obu+20].

— David Bruggemann, Menelaos Kanakis, Stamatios Georgoulis, and Luc Van Gool, "Automated Search for Resource-Efficient Branched Multi-Task Networks", BMVC, 2020 [Bru+20].

— Suman Saha, Wenhao Xu, Menelaos Kanakis, Stamatios Georgoulis, Yuhua Chen, Danda Pani Paudel, and Luc Van Gool, "Domain Agnostic Feature Learning for Image and Video Based Face Anti-Spoofing", CVPRW, 2020 [Sah+20].

* Equal contribution.

— David Bruggemann, Menelaos Kanakis, Anton Obukhov, Stamatios Georgoulis, and Luc Van Gool, "Exploring Relational Context for Multi-Task Dense Prediction", ICCV, 2021 [Bru+21].

— Suman Saha*, Anton Obukhov*, Danda Pani Paudel, Menelaos Kanakis, Yuhua Chen, Stamatios Georgoulis, and Luc Van Gool, "Learning to Relate Depth and Semantics for Unsupervised Domain Adaptation", CVPR, 2021 [Sah+21].

— Guolei Sun, Thomas Probst, Danda Pani Paudel, Nikola Popović, Menelaos Kanakis, Jagruti Patel, Dengxin Dai, and Luc Van Gool, "Task Switching Network for Multi-task Learning", ICCV, 2021 [Sun+21].

* Equal contribution.

CONTENTS

1

INTRODUCTION

The automatic visual perception and understanding of the physical world, through the use of computers, is a fundamental goal in computer vision research. Inspired by the human visual system, computer vision research aims to build algorithms that can utilize visual inputs, such as images, and enable machines to gain a high-level understanding of the visual input's content. To that extent, research is focused on the automatic extraction, analysis and understanding of important and useful information.

Initial attempts to computer vision date back to the work of Lawrence Roberts [Rob63], however, it was not until the seminal work of David Marr that the field observed significant improvements [Mar76; Mar82]. Marr's framework follows a bottom-up approach to scene understanding, where low-level cues, such as corners and edges, are utilized as building blocks towards the goal of attaining higher-level information. One of the earliest and most prominent example of this framework is the coupling of the Canny edge detector [Can86] with the Hough transform [Bal81] to obtain shape information, such as lines and circles. The continuous success of utilizing low-level cues on a wide variety of tasks such as stereo matching [Mor81], motion tracking [HS+88; Har93], image matching [Zha+95] and image retrieval [SM97], sparked the interest for more powerful and descriptive low-level features. Some of the most well known handcrafted feature extractors include SIFT [Low04], HOG [DT05], and SURF [BTG06]. Combining these features with machine learning methods, like SVM [CV95], has made the more challenging high-level task of image classification feasible [Csu+04; SWP05]. Nevertheless, the manually designed nature of such features make them sub-optimal when the design assumptions do not hold.

Inspired by the limitation of handcrafted feature extractors, Deep Neural Networks (DNNs) aim to jointly learn the bottom-up feature extractor and the prediction head, such as the classifier, by directly optimizing for the desired behaviour [LBH15]. DNNs are based on a composition of linear functions, non-linear activation functions, and pooling operations. These models are optimized using a cost function that captures the desired output behaviour, such as cross-entropy for classification, and large scale datasets. The computer vision community has heavily

adopted the use of DNNs, in particular Convolutional Neural Networks (CNNs) [LeC+89], since Krizhevsky et al. [KSH12] won the ImageNet classification challenge [Rus+15], outperforming the methods utilizing traditional handcrafted feature extractors by a large margin. Since then, CNNs have not only been utilized to improve image classification [SZ15; Sze+15; Sze+16; He+16; ZK16], but also a wide range of tasks. Such tasks include, but are not limited to, semantic segmentation [YK16a; Che+17; Zha+17; Yu+18; Che+18a], human pose estimation [NYD16; Cao+17; Sun+19; Cao+19], monocular depth estimation [Zho+17; Fu+18; God+19], object detection [Gir+14; Gir15; Ren+15; Red+16], and visual object tracking [Ber+16; Bha+19].

In an attempt to improve the representational capabilities of CNNs, networks became deeper [SZ15; He+16], wider [ZK16; Sun+19], or even replaced the convolutional operations with more descriptive alternatives [Dos+21; Tol+21]. We depict the progress on the ImageNet classification benchmark [Rus+15] over time in Fig. 1.1 [a]. As seen, consistent performance improvements have been reported over the years, however, these advances often come at the cost of increased computational complexity, such as the number of parameters (Fig. 1.2a) and the number of FLOPs (Fig. 1.2b). To an extent, these advances primarily require high-end Graphics Processing Units (GPUs) and Tensor Processing Units (TPUs), often found on cloud servers.

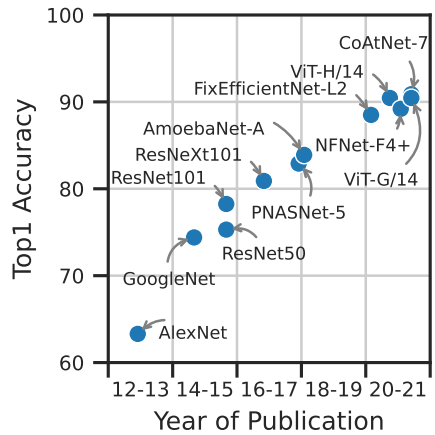
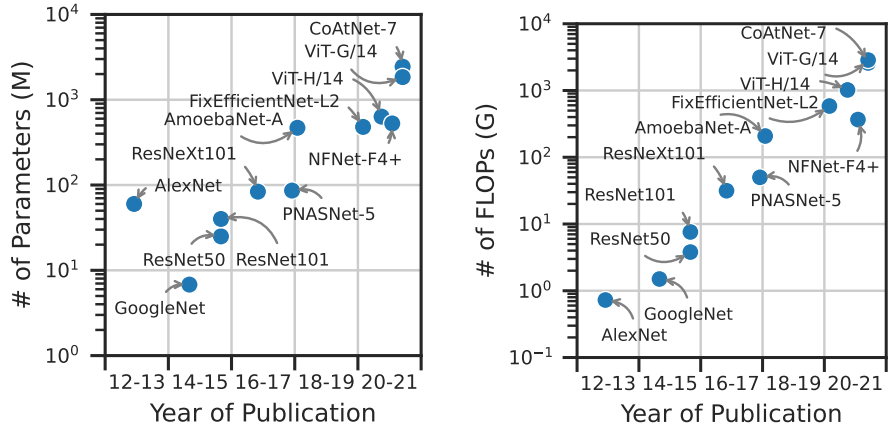


Figure 1.1: Progress on the ImageNet classification benchmark [Rus+15] over time, as reported on [PWC]. Consistent improvements can be observed over the years.

a We utilize the benchmark of [PWC], and report the best performing models that include Top1 accuracy, number of parameters and number of Floating-Point Operations (FLOPs). Specifically, we report the performance of AlexNet [KSH12], GoogLeNet [Sze+15], ResNet50/101 [He+16], ResNeXt101 [Xie+17], PNASNet-5 [Liu+18a], AmoebaNet-A [Rea+19], FixEfficientNet-L2 [Tou+19], ViT-H/14 [Dos+21], NFNet-F4+ [Bro+21], CoAtNet-7 [Dai+21], ViT-G/14 [Zha+22].



(a) Number (#) of parameters over time (b) Number (#) of FLOPs over time

Figure 1.2: We depict the change to the number of parameters, in Millions (M), and the number of Giga (G) FLOPs over time for the models in Fig. 1.1. As seen, in an attempt to increase the performance, the number of parameters and FLOPs consistently increases as well.

The great advancement in DNNs has further sparked interest in their use in robotics, Augmented Reality (AR), Virtual Reality (VR), self-driving cars, Internet of Things (IoT), and mobile phones [Sar+22]. However, cloud computing limitations prohibit its use for inference in such applications. Firstly, an unstable or lost internet connection makes the use of cloud processing impossible. Secondly, the prohibition of processing and storing sensitive data under data protection reculations, such as the General Data Protection Regulation (GDPR) [Cus+19]. Finally, as the number of users for any device or service increases, cloud servers are required to address the increased data transmission to and from the devices, as well as the increased processing demand, making it infeasible and cost-ineffective. To alleviate these problems, the aforementioned applications rely on on-board processing, referred to as *edge computing*. The on-board processing of DNNs addresses all of the limitations of cloud computing and has the potential to provide a deterministic and real-time experience [DD17]. However, unlike cloud servers, mainframes and workstations, embedded platforms have limited storage, memory, computational power, battery life, and often require faster and smaller software updates. These limitations can be, at part, addressed by a combination of the following methods, depending on the device-specific constraints:

Topological optimizations Topological optimizations aim to improve accuracy-per-operation or accuracy-per-parameter by changing the network's architecture. Notable examples include MobileNets [How+17; San+18; How+19], ShuffleNets [Zha+18a; Ma+18], EfficientNets [TL19; TL21], amongst others [Gho+18; Hua+18; Zop+18; Liu+18a; LSY18; Rad+20]

Hardware-aware optimizations Embedded platforms commonly provide limited or even no support for Full Precision (FP) arithmetics. Furthermore, they are often optimized to execute SIMD (Single Instruction, Multiple Data) Integer (Int) arithmetics [Ign+18]. While standard deep-learning libraries utilize 32-bit FP representations [Pas+19; Mar+15], the need for Int representations calls for Quantized Neural Networks (QNNs). By replacing FP with Int operands, QNNs reduce their storage and memory requirements with respect to equivalent DNNs, while complex FP arithmetic can be replaced by simpler Int arithmetics. Due to these properties, QNNs can be executed at higher throughput (number of operations per cycle) and arithmetic intensity (number of arithmetic operations per memory transaction) [CBD15; KS15; Ras+16; LZP17; Zhu+17; Liu+18b; Jac18; Nag+19; LS20].

Knowledge distillation Starting from a large model, referred to as "teacher", the aim is to transfer the knowledge to a smaller model, referred to as "student", that is more deployment friendly [HVD15]. Specifically, this can include knowledge transfer between models of the same architecture family, from ResNet-101 [He+16] to ResNet-50 [He+16], but also different architectures, such as from ResNet-101 [He+16] to a MobileNet [How+17]. Knowledge distillation can be seen as a function matching between the two networks, and has demonstrated great results in practice [HVD15; Rom+15; TV17; MM18; CH19; SS20; Xie+20; Bey+22]

Model pruning and decomposition Motivated by the over-parameterization of DNNs, pruning methods aim to identify and eliminate the redundant operations from the network. This can include pruning independent neurons [Han+15; HMD16], but often entire filters for new kernel with a regular shape [Li+17; Gor+18; Yan+18]. Similar to pruning, decomposition methods replace the existing filters with low-rank approximations. This can be on the two dimensional filters [Den+14; JVZ14; Zha+15], or the CNNs weight tensors [Ose11; Nov+15; Su+18; Leb+15; Kim+16; Gar+16; WAA17; Wan+18a].

Multi-task learning The methods discussed so far focus on learning a single network per task. In a different direction, Multi-Task Learning (MTL) focuses on learning multiple tasks with a single network. MTL was originally proposed with the primary focus of improving the performance of a target task, by utilizing the training signal of additional related tasks as an inductive bias [Car97]. However, the bottom-up approach of neural networks has enabled the sharing of parameters and computing amongst different tasks, making them a great framework to not only increase task performance [Mis+16; Xu+18; Ran+19; Hoy+21; Bru+21] but also decrease the total number of parameters and FLOPs [Kok17; RBV17; BV17; RPC17; RBV18; MRK19; Bru+20; Sta+20].

1.1 THESIS OVERVIEW

The aim of this thesis is to design computationally efficient deep neural networks, such as improving performance given computational constraints or decreasing complexity with minor performance degradation. Specifically, the thesis is divided into the following chapters:

Chapter 2 Devices such as mobile phones must often be capable of making predictions for a number of different tasks. This calls for a large number of highly specialized single-task networks, increasing the total number of parameters to be stored. This can be alleviated by a multi-task network, however, two common challenges in developing multi-task models are often overlooked. First, enabling the model to be inherently incremental, continuously incorporating information from new tasks without forgetting the previously learned ones, referred to as incremental learning. Second, adverse interactions amongst tasks can significantly degrade the single-task performance in a multi-task setup. In Chapter 2, we demonstrate that both can be achieved simply by reparameterizing the convolutions of standard neural network architectures into a non-trainable shared part (filter bank), and task-specific parts (modulators), where each modulator has a fraction of the filter bank parameters. Thus, our reparameterization enables the model to learn new tasks without adversely affecting the performance of existing ones. This enables the more efficient use of the device's storage space. Furthermore, it also allows for faster and smaller software updates, where new tasks can simply be added through task-specific modulators while maintaining the existing filter bank. The content of this chapter is based on the following publication:

— Menelaos Kanakis, David Bruggemann, Suman Saha, Stamatios Georgoulis, Anton Obukhov, and Luc Van Gool, "Reparameterizing Convolutions for Incremental Multi-Task Learning without Task Interference", ECCV, 2020 [Kan+20].

Chapter 3 Improving accuracy-per-operation or accuracy-per-parameter is key when dealing with limited computational constraints. While topological optimizations aim to do so through the design of novel architectures, MTL utilizes a fixed architecture and aims to improve model performance and generalization on a target task by jointly optimizing it with an auxiliary task. However, the current practice requires additional labeling efforts for the auxiliary task, while not guaranteeing better model performance. In Chapter 3, we investigate and find that jointly training a dense prediction (target) task with a self-supervised (auxiliary) task can consistently improve the performance of the target task, while eliminating the need for labeling auxiliary tasks. We refer to this joint training as Composite Learning (CompL). Experiments of CompL on monocular depth estimation, semantic segmentation, and boundary detection show consistent performance improvements in fully and partially labeled datasets. Further analysis on depth estimation reveals that joint training with self-supervision outperforms most labeled auxiliary tasks. We also find that CompL can improve model robustness when the models are evaluated in new domains. This enables the improvement of the accuracy-per-operation and accuracy-per-parameter Pareto fronts without the need of adapting the neural network architecture deployed. The content of this chapter is based on the following publication:

— Menelaos Kanakis, Thomas E. Huang, David Bruggemann, Fisher Yu, and Luc Van Gool, "Composite Learning for Robust and Effective Dense Predictions", WACV, 2023 [Kan+23].

Chapter 4 Similar to Fig. 1.2, the design of more complex and powerful neural network models has significantly advanced the state-of-the-art in visual object tracking. However, in the pursuit of increased tracking performance, runtime is often hindered, while efficient tracking architectures have received surprisingly little attention. In Chapter 4, we introduce the Exemplar Transformer, a transformer module utilizing a single instance level attention layer for realtime visual object tracking. E.T.Track, our visual tracker that incorporates Exemplar Transformer modules, runs at 47 *FPS* on a CPU. This is up to 8× faster than other transformer-based models. When compared to lightweight trackers that can operate in

realtime on standard CPUs, E.T.Track consistently outperforms all other methods. The content of this chapter is based on:

— Philippe Blatter*, Menelaos Kanakis*, Martin Danelljan, and Luc Van Gool, "Efficient Visual Tracking with Exemplar Transformers", WACV, 2023 [Bla+23][1].

Chapter 5 Computationally limited platforms such as robots, mobile, and AR devices rely on efficient and accurate localization and mapping in 3D space. Such systems still rely on traditional hand-crafted methods for efficient generation of lightweight descriptors, a common limitation of the more powerful neural network models that come with high computing and specific hardware requirements. In Chapter 5, we focus on the adaptations required by detection and description neural networks to enable their use in computationally limited platforms. To that end, we investigate and adapt network quantization techniques to accelerate inference and enable its use on compute limited platforms. In addition, we revisit common practices in descriptor quantization and propose the use of a binary descriptor normalization layer, enabling the generation of distinctive binary descriptors with a constant number of ones. ZippyPoint, our efficient quantized network with binary descriptors, improves the network runtime speed, the descriptor matching speed, and the size of sparse 3D models, by at least an order of magnitude when compared to full-precision counterparts. These improvements come at a minor performance degradation. The content of this chapter is based on the following pre-print:

— Menelaos Kanakis*, Simon Maurer*, Matteo Spallanzani, Ajad Chhatkuli, and Luc Van Gool, "ZippyPoint: Fast Interest Point Detection, Description, and Matching through Mixed Precision Discretization", arXiv preprint arXiv:2203.03610, 2022 [Kan+22][2].

* Equal contribution.
[1] The primary technical contributions of Kanakis M. include the idea and first implementation of the Exemplar Attention layer.
[2] The primary technical contributions of Kanakis M. include the idea and implementation of the Binary Normalization (Bin.Norm) layer, and the layer partitioning and traversal strategy.

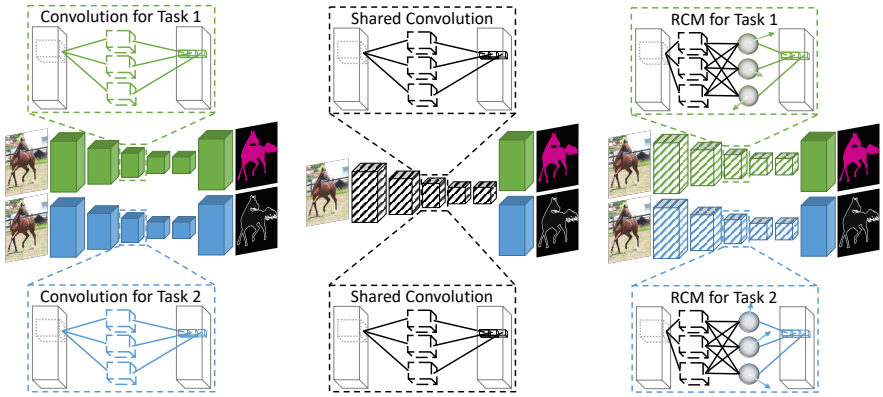
2

REPARAMETERIZING CONVOLUTIONS FOR MULTI-TASK LEARNING

Multi-task networks are commonly utilized to alleviate the need for a large number of highly specialized single-task networks. However, two common challenges in developing multi-task models are often overlooked in literature. First, enabling the model to be inherently incremental, continuously incorporating information from new tasks without forgetting the previously learned ones (incremental learning). Second, eliminating adverse interactions amongst tasks, which has been shown to significantly degrade the single-task performance in a multi-task setup (task interference). In this chapter, we show that both can be achieved simply by reparameterizing the convolutions of standard neural network architectures into a non-trainable shared part (filter bank) and task-specific parts (modulators), where each modulator has a fraction of the filter bank parameters. Thus, our reparameterization enables the model to learn new tasks without adversely affecting the performance of existing ones. The results of our ablation study attest the efficacy of the proposed reparameterization. Moreover, our method achieves state-of-the-art on two challenging multi-task learning benchmarks, PASCAL-Context [Mot+14] and NYUD [Sil+12], and also demonstrates superior incremental learning capability as compared to its close competitors.

2.1 INTRODUCTION

Over the last decade, CNNs have been established as the standard approach for many computer vision tasks, like image classification [KSH12; SZ15; He+16], object detection [Gir+14; Red+16; Liu+16], semantic segmentation [LSD15; Che+17; Zha+17], and monocular depth estimation [EPF14; Lai+16]. Typically, these tasks are handled by CNNs independently, i.e., a separate model is optimized for each task, resulting in several task-specific models (Fig. 2.1a). However, real-world problems are more complex and require models to perform multiple tasks on-demand without significantly compromising each task's performance. For example, an interactive advertisement system tasked with displaying targeted content to its audience should be able to detect the presence



(a) Single-Task setup (b) Multi-Task setup (c) RCM setup (ours)

Figure 2.1: (a) Optimizing independent models per task allows for the easy addition of new tasks, at the expense of a multiplicative increase in the total number of parameters with respect to a single model (green and blue denote task-specific parameters). (b) A single backbone for multiple tasks must be meaningful to all, thus, all tasks interact with the said backbone (black indicates common parameters). (c) Our proposed setup, RCM (Reparameterized Convolutions for Multi-task learning), uses a pre-trained filter bank (depicted in black) and independently optimized task-specific modulators(depicted in blue or green) to adapt the filter bank on a per-task basis. New task addition is accomplished by training the task-specific modulators, thus explicitly addressing task interference, while the additional parameters required are a fraction of those needed for completely task-specific convolutions.

of humans in its viewpoint effectively, estimate their gender and age group, recognize their head pose, etc. At the same time, there is a need for flexible models able to gradually add more tasks to their knowledge, without forgetting previously known tasks or having to re-train the whole model from scratch. For instance, a car originally deployed with lane and pedestrian detection functionalities can be extended with depth estimation capabilities post-production.

When it comes to learning multiple tasks under a single model, MTL techniques [Car97; Rud17] have been employed in the literature. On the one hand, encoder-focused approaches [Mis+16; Kok17; Lu+17; DZ17; Nev+17; LJD19; Bra+19; Van+20; Bru+20] emphasize learning feature representations from multi-task supervisory signals by employing architectures that encode shared and task-specific information. On the other hand, decoder-focused approaches [Xu+18; Zha+18b; Zha+19d;

VGG20; Bru+21] utilize the multi-task feature representations learned at the encoding stage to distill cross-task information at the decoding stage, thus refining the original feature representations. In both cases, however, the joint learning from multiple supervisory signals (i.e., tasks) can hinder the individual task performance if the associated tasks point to conflicting gradient directions during the update step of the shared feature representations (Fig. 2.1b). Formally this is known as *task interference* or *negative transfer* and has been well documented in the literature [Kok17; Zha+18c; MRK19]. To suppress negative transfer, several approaches [Che+18b; KGC18; Sin+18; Guo+18; Zha+18c; SK18; MRK19] dynamically re-weight each task's loss function or re-order the task learning, to find a 'sweet spot' where individual task performance does not degrade significantly. Arguably, such approaches mainly focus on mitigating the negative transfer problem in the MTL architectures above, rather than eliminating it, as demonstrated in Sec. 2.3.2. At the same time, existing works seem to disregard the fact that MTL models are commonly desired to be incremental, i.e., information from new tasks should be continuously incorporated while existing task knowledge is preserved. In existing works, the MTL model has to be re-trained from scratch if the task dictionary changes; this is arguably sub-optimal.

Recently, task-conditional networks [MRK19] emerged as an alternative for MTL, inspired by work in multi-domain learning [BV17; RBV17; RBV18]. That is, performing separate forward passes within an MTL model, one for each task, every time activating a set of task-specific responses on top of the shared responses. In practice, task-conditional networks behave similar to single task networks. Unlike single task networks, the addition of a new task can be accomplished with a smaller increase in the total number of parameters since, rather than requiring a new single task model, it instead only required the task-specific modulators. This makes task-conditional networks a candidate for use on computational limited platforms where the available storage space cannot hold all single task models due to a large total number of parameters. However, the proposed architecture in [MRK19] is prone to task interference due to the inherent presence of shared modules, which is why the authors introduced an adversarial learning scheme on the gradients to minimize the performance degradation. Moreover, the model needs to be trained from scratch if the task dictionary changes, limiting its use for Incremental Learning (IL).

Existing works primarily focus on either improving the multi-task model's performance, or reducing the number of parameters and computations in the MTL model. In this chapter, we take a different route and

explicitly tackle the problems of IL and task interference in MTL. We show that both problems can be addressed simply by reparameterizing the convolutional operations of a neural network. In particular, building upon the task-conditional MTL direction, we propose to decompose each convolution into a shared part that acts as a filter bank encoding common knowledge, and task-specific modulators that adapt this common knowledge uniquely for each task. Fig. 2.1c illustrates our approach, Reparameterized Convolutions for Multi-task learning (RCM). Unlike existing works, the shared part in our case is not trainable to explicitly avoid negative transfer. Most notably, as any number of task-specific modulators can be introduced in each convolution, our model can incrementally solve more tasks without interfering with the previously learned ones. Our results demonstrate that the proposed RCM can outperform state-of-the-art methods in MTL and IL. Furthermore, we address the common MTL challenge of task interference by construction. Specifically, by ensuring tasks are able to only optimize their task-specific components and are unable to interact with the other tasks.

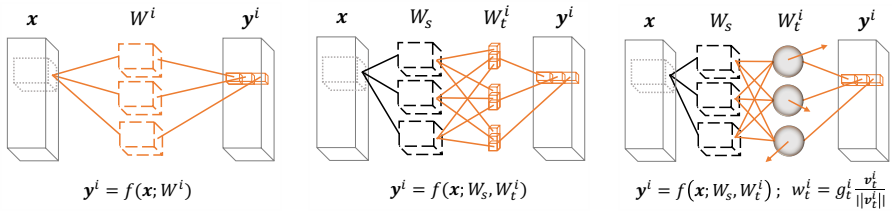
2.2 RELATED WORK

Multi-Task Learning (MTL) aims at developing models that can solve a multitude of tasks [Car97; Rud17]. In computer vision, MTL approaches can roughly be divided into encoder-focused and decoder-focused ones. Encoder-focused approaches primarily emphasize on architectures that can encode multi-purpose feature representations through supervision from multiple tasks. Such encoding is typically achieved, for example, via feature fusion [Mis+16], branching [Kok17; Nev+17; Lu+17; Van+20; Bru+20], self-supervision [DZ17], attention [LJD19], or filter grouping [Bra+19]. Decoder-focused approaches start from the feature representations learned at the encoding stage, and further refine them at the decoding stage by distilling information across tasks in a one-off [Xu+18], sequential [Zha+18b], recursive [Zha+19d], multi-scale [VGG20], or even with the use of more powerful transformer layers [Bru+21]. Due to the inherent layer sharing, the approaches above typically suffer from task interference. Several works proposed to dynamically re-weight the loss function of each task [Che+18b; KGC18; Sin+18; SK18], sort the order of task learning [Guo+18], or adapt the feature sharing between *related* and *unrelated* tasks [Zha+18c], to mitigate the effect of negative transfer. In general, existing MTL approaches have primarily focused on improving multi-task performance or reducing the network parameters and com-

putations. Instead, in this chapter, we look at the largely unexplored problems of incremental learning and negative transfer in MTL models and propose a principled way to tackle them, while additionally reducing the total number of parameters.

Incremental Learning (IL) is a paradigm that attempts to augment the existing knowledge by optimizing on new data. IL is often used, for example, when aiming to add new classes [Reb+17] to an existing model, or learn new domains [LH17]. It aims to mitigate 'catastrophic forgetting' [Fre99], the phenomenon of forgetting old tasks as new ones are learned. To minimize the loss of existing knowledge, Li et al. [LH17] optimized the new task while preserving the old task's responses. Other works [Kir+17; Lee+17] constrained the optimization process to minimize the effect learning has on weights important for older tasks. Rebuffi et al. [Reb+17] utilized exemplars that best approximate the mean of the learned classes in the feature space to preserve performance. Note that the performance of such techniques is commonly upper bounded by the joint training of all tasks. More relevant to our work, in a multi-domain setting, a few approaches [RBV17; RBV18; RT18; MDL18] utilize a pre-trained network that remains untouched, and instead learn domain-specific components that adapt the behavior of the network to address the performance drop common in IL techniques. Inspired by this research direction, we investigate the training of parts of the network, while keeping the remaining components constant from initialization amongst all tasks. This technique not only addresses catastrophic forgetting but also task interference, which is crucial in MTL.

Decomposition of filters and tensors within CNNs has been explored in the literature primarily in an attempt to compress neural networks, or reducing their inference time. Techniques utilized include filter-wise decomposition into a product of low-rank filters [JVZ14], filter groups [Pen+18], a basis of filter groups [Li+19b], amongst others. In contrast, tensor-wise examples include SVD decomposition [Den+14; Zha+15], CP-decomposition [Leb+15], Tucker decomposition [Kim+16], Tensor-Train decomposition [Ose11], Tensor-Ring decomposition [Zha+16], T-Basis [Obu+20], etc. Instead, we decompose each convolutional operation into two components: a shared and a task-specific part. Note that although we utilize the SVD decomposition for simplicity, the same principles hold for other decomposition types too.



(a) Standard Conv. (b) RC without NFF (c) RC with NFF

Figure 2.2: **(a)** A standard convolutional module for a given task i, with task-specific weights W^i in orange. **(b)** A Reparameterized Convolution (RC) consisting of a shared filter bank W_s in black, and task-specific modulator W_t^i in orange. **(c)** An RC with Normalized Feature Fusion (NFF), consisting of a shared filter bank W_s in black, and task-specific modulator W_t^i in orange. Each row $\boldsymbol{w}_t^i$ of W_t^i is reparameterized as $g_t^i \cdot \boldsymbol{v_t^i} / \parallel \boldsymbol{v_t^i} \parallel$.

2.3 REPARAMETERIZING CNNS FOR MULTI-TASK LEARNING

In this section, we present techniques to adapt a convolutional operation, such that the CNN can increasingly learn new tasks in an MTL setting while scaling more efficiently than the alternative of optimizing single-task models. Sec. 2.3.1 introduces the problem formulation. Sec. 2.3.2 discusses the adverse effect of task interference in MTL and motivates the importance of CNN reparameterization. Sec. 2.3.3 presents techniques to reparameterize convolutional operations and limit the parameter increase with respect to task-specific models.

2.3.1 *Problem Formulation*

Given P tasks and input tensor $\boldsymbol{x}$, we aim to learn a function $f(\boldsymbol{x}; W_s, W_t^i) = \boldsymbol{y}^i$ that holds for task $i = 1, 2, \ldots P$, where W_s and W_t^i are the shared and task-specific parameters, respectively. Unlike existing approaches [Lu+17; Mis+16] which learn such functions $f(\cdot)$ on the layer level of the network, i.e., explicitly designing shared and task-specific layers, we aim to learn f on a block-level by *reparameterizing* the convolutional operation, and adapting its behaviour conditioned on the task i, as depicted in Fig. 2.2b and Fig. 2.2c. By doing so, we can explicitly address the task interference and catastrophic forgetting problems within an MTL setting.

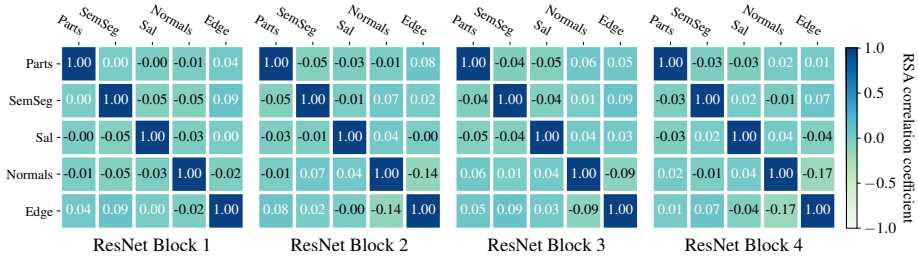


Figure 2.3: Visualization of the Representation Similarity Analysis (RSA) on the task-specific gradients at different depths of a ResNet-26 model [MRK19]. The analysis was conducted on: human parts segmentation (Parts), semantic segmentation (SemSeg), saliency estimation (Sal), normals estimation (Normals), and edge detection (Edge).

2.3.2 *Task Interference*

To motivate the importance of addressing task interference by construction, we analyze the task-specific gradient directions on the shared modules of a state-of-the-art MTL model. Specifically, we utilize the work of [MRK19], who used a discriminator to enforce indistinguishable gradients amongst tasks.

We acquire the gradients from the training dataset of PASCAL-Context [Mot+14] for each task, using minibatches of size 128, yielding 40 minibatches. We then use the Representation Similarity Analysis (RSA), proposed in [DR19] for transfer learning, as a means to quantify the correlation of the gradients amongst the different tasks. Fig. 2.3 depicts the task gradient correlations at different depths of a ResNet-26 model [He+16], trained to have indistinguishable gradients in the output layer [MRK19]. It can be seen that there is a limited gradient correlation amongst the tasks, demonstrating that addressing task interference indirectly (here with the use of adversarial learning on the gradients) is a very challenging problem. We instead follow a different direction and propose to utilize reparameterizations with shared components amongst different tasks that are untouched during the training process, and each task being able to optimize only its task-specific parameters. As such, task interference is eliminated by construction.

2.3.3 *Reparameterizing Convolutions*

We define a convolutional operation $f(\boldsymbol{x}; \boldsymbol{w}) = y$ for the single-task learning setup, depicted in Fig. 2.2a. $\boldsymbol{w} \in \mathbb{R}^{k^2 c_{in}}$ denotes the parameters of a single convolutional layer with a kernel size k and c_{in} channels. We omit the bias to simplify notation. $\boldsymbol{x} \in \mathbb{R}^{k^2 c_{in}}$ is the input tensor volume at a given spatial location, and y is the scalar response. Note that both $\boldsymbol{x}$ and $\boldsymbol{w}$ are expressed in vector notation. Assuming c_{out} such filters, the convolutional operator can be rewritten in matrix notation as $f(\boldsymbol{x}; W) = \boldsymbol{y}$, where $\boldsymbol{y} \in \mathbb{R}^{c_{out}}$ provides c_{out} responses, and $W \in \mathbb{R}^{c_{out} \times k^2 c_{in}}$. In a single-task setup:

$$f(\boldsymbol{x}; W^1) = \boldsymbol{y}^1, \ \ldots \ , \ f(\boldsymbol{x}; W^P) = \boldsymbol{y}^P \tag{2.1}$$

where W^i and $\boldsymbol{y}^i$ are the task-specific parameters and responses for a given convolutional layer, respectively. The total number of parameters for this setup is $Pk^2 c_{in} c_{out}$. Our goal is to reparameterize $f(\cdot)$ in Eqn. 2.1 as:

$$f(\boldsymbol{x}; W^i) = h(\boldsymbol{x}; W_s, W_t^i), \quad \forall i = 1, \ldots, P \tag{2.2}$$

using a set of shared ($W_s \in \mathbb{R}^{c_{out} \times k^2 c_{in}}$) and task-specific ($W_t^i \in \mathbb{R}^{c_{out} \times c_{out}}$) parameters for each convolutional layer of the backbone. Our formulation aims to retain the prediction performance of the original convolutional layer (Eq. 2.1), while simultaneously reducing the rate in which the total number of parameters grows. The total number of parameters now becomes $(k^2 c_{in} + P c_{out}) c_{out}$, which is less than $Pk^2 c_{in} c_{out}$ for standard layers. We argue that this reparameterization is necessary for coping with task interference and incremental learning in an MTL setup, in which we only optimize for task-specific parameters W_t^i, while keeping the shared parameters W_s intact. Note that, when adding a new task $i = \omega$, we do not need to train the entire network from scratch as in [MRK19]. We only optimize W_t^ω for each layer of the reparameterized CNN.

We denote our reparameterized convolutional layer as a matrix multiplication between the two sets of parameters: $W_t^i W_s$. In order to find a set of parameters $W_t^i W_s$ that approximates the single-task weights W^i a natural choice is to minimize the Frobenius norm $\|W_t^i W_s - W^i\|_F$ directly. Even though direct minimization of this metric is appealing due to its simplicity, it poses some major caveats. Firstly, it assumes all directions in the parameter space affect the final performance for task i in the same way and are thus penalized uniformly. However, two different solutions for W_t^i with the same Frobenius norm can yield drastically different

losses. Secondly, this approximation is performed independently for each convolutional layer, neglecting the chain effect an inaccurate prediction in one layer can have in the succeeding layers. In the remainder of this section, we propose a different technique to address these limitations.

Reparameterized Convolution (RC) We implement the Reparameterized Convolution (RC) $W_t^i W_s$ as a stack of two 2D convolutional layers without a non-linearity in between, with W_s having a spatial filter size k and W_t^i being a 1×1 convolution (Fig. 2.2b). We optimize only W_t^i directly on the task-specific loss function using stochastic gradient descent while keeping the shared weights W_s constant[a]. This ensures that training for one task is independent of other tasks, ruling out interference amongst tasks while optimizing the metric of interest.

Normalized Feature Fusion (NFF) One can view $\boldsymbol{w}_t^i$, a row in matrix W_t^i, as a soft filter adaptation mechanism, i.e., a modulator which generates new task-specific filters from a given filter bank W_s, depicted in Fig. 2.2b. However, instead of training the vector $\boldsymbol{w}_t^i$ directly, we propose its reparameterization into two terms, a vector term $\boldsymbol{v}_t^i \in \mathbb{R}^{c_{out}}$, and a scalar term g_t^i as:

$$\boldsymbol{w}_t^i = g_t^i \frac{\boldsymbol{v}_t^i}{\| \boldsymbol{v}_t^i \|}, \tag{2.3}$$

where $\| \cdot \|$ denotes the Euclidean norm. We refer to this reparameterization as Normalized Feature Fusion (NFF), depicted in Fig. 2.2c. NFF provides an easier optimization process in comparison to an unconstrained $\boldsymbol{w}_t^i$. This reparametrization enforces $\boldsymbol{v}_t^i / \| \boldsymbol{v}_t^i \|$ to be unit length and point in the direction which best merges the filter bank. The vector norm $\| \boldsymbol{w}_t^i \| = g_t^i$ learns independently the appropriate scale of the newly generated filters, and thus the scale of the activation. Directly optimizing $\boldsymbol{w}_t^i$ attempts to learn both jointly, which is a harder optimization problem. Normalizing weight tensors has been generally explored for speeding up the convergence of the optimization process [Dau+17; SK16; SS05]. In our work, we use it differently and demonstrate empirically that such a reparameterization in series with a filter bank also improves performance in the MTL setting. As seen in Eq. 2.3, $\boldsymbol{w}_t^i$ is replaces by g_t^i and $\boldsymbol{v}_t^i$, however, $\boldsymbol{w}_t^i$ can be computed after training and used directly for deployment, eliminating additional overhead.

a To ensure compliance with ImageNet [Den+09] initialization, the new architecture is first pre-trained on ImageNet using the publicly available training script from PyTorch [Pas+19].

Response Initialization (RI) We build upon the findings of matrix/tensor decomposition literature [Den+14; Zha+15] that network weights/responses lie on a low dimensional subspace. We further assume that such a subspace can be beneficial for multiple tasks, and thus good for network initialization under a MTL setup. To this end, we identify a meaningful subspace of the responses for the generation of a better filter bank W_s when compared to that directly learned by pre-training W_s on ImageNet.

Formally, let $\boldsymbol{y} = f(\boldsymbol{x}; W^m) = W^m \boldsymbol{x}$ be the responses for input tensor $\boldsymbol{x}$, where $W^m \in \mathbb{R}^{c_{out} \times k^2 c_{in}}$ are the pre-trained ImageNet weights. We define $Y \in \mathbb{R}^{c_{out} \times n}$ as a matrix containing n responses of $\boldsymbol{y}$ with the mean vector $\overline{\boldsymbol{y}}$ subtracted. We compute the eigen-decomposition of the covariance matrix $YY^T = USU^T$ using Singular Value Decomposition (SVD), where $U \in \mathbb{R}^{c_{out} \times c_{out}}$ is an orthogonal matrix with the eigenvectors on the columns, and S is a diagonal matrix of the corresponding eigenvalues. Using the projection to (U^T) and from (U) the latent space, we can rewrite $\boldsymbol{y} = UU^T(\boldsymbol{y} - \overline{\boldsymbol{y}}) + \overline{\boldsymbol{y}}$. Note that the centering operation is important since SVD assuming centered responses. When introduced in our standard convolution operation it yields,

$$\begin{aligned} \boldsymbol{y} &= W^m \boldsymbol{x} = UU^T(\boldsymbol{y} - \overline{\boldsymbol{y}}) + \overline{\boldsymbol{y}} \\ \boldsymbol{y} &= UU^T(W^m \boldsymbol{x} - \overline{\boldsymbol{y}}) + \overline{\boldsymbol{y}} \\ \boldsymbol{y} &= UU^T W^m \boldsymbol{x} + (\overline{\boldsymbol{y}} - UU^T \overline{\boldsymbol{y}}) \\ \boldsymbol{y} &= W_t^i W_s \boldsymbol{x} + b. \end{aligned} \tag{2.4}$$

Through this formulation, the non-trainable shared filter bank W_s is initialized as $U^T W^m$, and implemented as a $k \times k$ convolution, with k being the filter size of W^m. The task-specific modulators W_t^i are instead initialized with U, and implemented as a 1×1 convolution. Finally, the bias b can be added to the running mean of the batchnorm layer following the convolution [IS15]. We refer to this initialization methodology as Response Initialization (RI).

2.4 EXPERIMENTS

2.4.1 *Datasets*

We focus our evaluation on dense prediction tasks, making use of two datasets. We conduct the majority of the experiments on PASCAL [Eve+10], and more specifically, PASCAL-Context [Mot+14]. We address

edge detection (Edge), semantic segmentation (SemSeg), human parts segmentation (Parts), surface normals estimation (Normals), and saliency (Sal). We evaluate single-task performance using optimal dataset F-measure (odsF) [MFM04] for edge detection, mean intersection over union (mIoU) for semantic segmentation, human parts and saliency, and finally mean error (mErr) for surface normals. Labels for human parts segmentation are acquired from [Che+14], while for saliency and surface normals from [MRK19].

We further evaluate the proposed method on the smaller NYUD dataset [Sil+12], comprised of indoor scenes, on edge detection (Edge), semantic segmentation (SemSeg), surface normals estimation (Normals), and depth (Depth). The evaluation metrics for edge detection, semantic segmentation, and surface normals estimation are identical to those for PASCAL-Context, while for depth we use root mean squared error (RMSE).

2.4.2 *Architecture*

All of our experiments make use of the DeepLabv3+ architecture [Che+18a], originally designed for semantic segmentation, which performs competitively for all tasks of interest as demonstrated in [MRK19]. DeepLabv3+ encodes multi-scale contextual information by utilizing a ResNet [He+16] encoder with a-trous convolutions [Che+17] and an a-trous spatial pyramid pooling (ASPP) module, while a decoder with a skip connection refines the predictions. Unless otherwise stated, we use a ResNet-18 (R-18) based DeepLabv3+, and report the mean performance of five runs for each experiment.

2.4.3 *Implementation Details*

We follow closely the implementation details of [MRK19], listed below for completeness.

Generic hyperparamaters All models are optimized using SGD with a learning rate 0.005, momentum 0.9, weight decay 0.0001, and the "poly" learning rate schedule [Che+17]. We use a single GPU with a minibatch of 8 images. The input images during training are augmented with random horizontal flips and random scaling in the range [0.5, 2.0] in 0.25 increments. The validity of these hyperparameters has already been tested in [MRK19], and hence they are used in all our experiments too, in order to ensure fair comparisons amongst different methods.

Dataset specific hyperparameters PASCAL-Context [Mot+14] models are trained for 60 epochs. The spatial size of the input images is 512×512. NYUD [Sil+12] models are trained for 200 epochs. The spatial size of the input images is 425×560. Images of insufficient size are padded with the mean color.

Task weighting and loss functions As is common in MTL, losses require careful loss weighting [MRK19; VGG20; KGC18; SK18], where each loss is task-dependent. For edge detection, we optimize the binary cross-entropy (BCE) loss, scaled by 50. Due to the class imbalance between the edge and non-edge pixels, edge pixels are penalized with a weight 0.95, while non-edge pixels with a scale of 0.05, accommodating [Kok16; Man+17]. For evaluation, we set the maximum allowed mislocalization of the optimal dataset F-measure (odsF) [MFM04] to 0.0075 and 0.011 for PASCAL-Context and NYUD, respectively, using the package of [PM15]. Semantic segmentation and human parts segmentation are optimized with cross-entropy loss, weighted by the factors of 1 and 2, respectively. Predictions of surface normals, normalized to unit length, and depth modalities, are penalized using the $\mathcal{L}_1$ loss, scaled by 10 and 1 respectively. Saliency is optimized using the BCE loss, weighted by a factor of 5.

2.4.4 *Evaluation Metric*

We follow standard practice [MRK19; VGG20] and quantify the performance of a model m as the average per-task performance drop with respect to the corresponding single-task baseline b:

$$\Delta_m = \frac{1}{P}\sum_{i=1}^{P}(-1)^{l_i}\frac{M_{m,i} - M_{b,i}}{M_{b,i}} \tag{2.5}$$

where l_i is either 1 or 0 if a lower or a greater value indicates better performance, respectively, for a performance measure M. P indicates the total number of tasks.

2.4.5 *Baseline*

To ensure our re-implementation provides a strong baseline, Table 2.1 compares the single-task performance of our implementation using a ResNet-18 based DeepLabv3+, the results from [VGG20] using a ResNet-18 based FPN [Lin+17], and the results from [MRK19] who utilized a

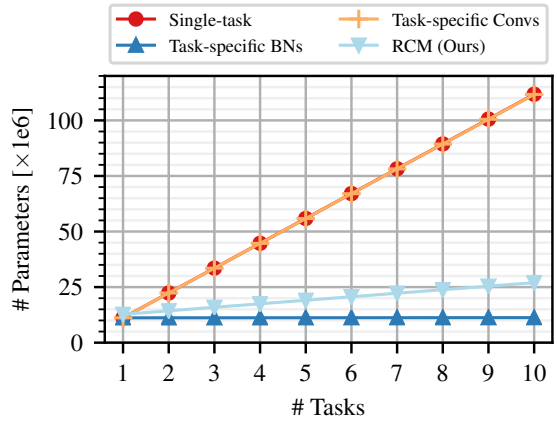


Figure 2.4: Total number of parameters with respect to the number of tasks for R-18 backbone. The total number of parameters when using RCM scale at a slower rate than the since-task alternative.

Table 2.1: We report the single-task performance of the baseline implementations of [MRK19; VGG20] for similar architectures on PASCAL-Context. The arrow indicates the direction for better performance. Bold indicates the best performance. Our implementation outperforms the others on every task.

Methods	Tasks				
	Edge ↑	SemSeg ↑	Parts ↑	Normals ↓	Sal ↑
ASTMT [MRK19]	70.30	63.90	55.90	15.10	63.90
MTI-Net [VGG20]	68.20	64.49	57.43	14.77	66.38
Ours	**71.88**	**66.22**	**59.69**	**13.64**	**66.62**

ResNet-26 based DeepLabv3+. We demonstrate that our single-task baseline outperforms both works on every task, and even though the numbers are not directly comparable due to minor implementation differences, it provides a strong baseline to conduct our study.

2.4.6 *Analysis of network module sharing*

We investigate the level of task-specific adaptation required for a common backbone to perform competitively to single-task models, while additionally eliminating negative transfer. In other words, the necessity for task-specific modules, i.e., convolutions (Convs) and batch normalizations (BNs) [IS15]. To this extent, we optimize for task-specific Convs, BNs, or both throughout the entire network. Modules that are not being

Table 2.2: We evaluate the effect optimizing task-specific Convs and BNs (✓) have on the performance of PASCAL-Context. The ✓ denotes modules optimized to be task-specific, while its absence indicates the use of the original ImageNet weights. Task-specific optimization of Convs is essential to achieve competitive performance to single-task networks, however, it comes at the cost of an approximately equal number of parameters. The arrow indicates the direction for better performance.

Modules		Tasks					
Convs	BNs	Edge ↑	SemSeg ↑	Parts ↑	Normals ↓	Sal ↑	$\Delta_m\%$ ↓
		67.32	60.37	47.86	17.40	58.39	14.98
	✓	69.80	63.93	53.22	14.78	64.44	5.76
✓		71.72	66.00	59.05	13.78	66.31	0.62
✓	✓	71.88	66.22	59.69	13.64	66.62	-

optimized maintain their ImageNet pre-trained parameters. Table 2.2 presents the effect on performance, while Fig. 2.4 depicts the total number of parameters with respect to the number of tasks. Experiments vary from common a frozen encoder with common Convs and BNs, to task-specific Convs and BNs, indicating the standard single-task baseline.

The model utilizing a common and unoptimized backbone pre-trained on ImageNet, as expected, is unable to perform competitively to the single-task counterpart, with a performance drop of 14.98%. Task-specific BNs significantly improve performance with a percentage drop of 5.76%, at a minimal increase in parameters, Fig. 2.4. The optimization of Convs is essential for competitive performance to single-task, with a percentage drop of 0.62%. However, the increase in parameters is comparable to single-task, which is undesirable, Fig. 2.4.

2.4.7 *Ablation study*

To validate the proposed methodology from Sec. 2.3, we conduct an ablation study, presented in Table 2.3. We additionally report the performance of a model trained jointly on all tasks, consisting of a fully shared encoder and task-specific decoders (Multi-task). This multi-task model is not trained in an IL setup but merely serves as a reference to the traditional multi-tasking techniques. We report a performance drop of 3.32% with respect to the single-task setup.

Table 2.3: Ablation experiments for the proposed Reparameterized Convolution when utilizing Response Initialization (RI), and or Normalized Feature Fusion (NFF) on PASCAL-Context dataset. The arrow indicates the direction for better performance.

		Tasks					
		Edge ↑	SemSeg ↑	Parts ↑	Normals ↓	Sal ↑	$\Delta_m\%$ ↓
Single-task		71.88	66.22	59.69	13.64	66.62	-
Multi-task		70.74	62.43	57.89	14.43	66.31	3.32
Reparameterized Convolution							
NFF	RI						
		71.10	64.56	56.87	13.91	66.37	2.13
✓		71.12	64.71	56.91	13.90	66.33	2.07
	✓	71.36	65.58	57.99	13.70	66.21	1.12
✓	✓	71.34	65.70	58.12	13.70	66.38	0.99

Reparameterized Convolution (RC) We first develop a new baseline for our proposed reparameterization, where we replace every convolution with the RC (Sec. 2.3.3) counterpart. As seen in Table 2.3, RC without NFF and RI achieves a performance drop of 2.13%, outperforming the 3.32% drop of the multi-task baseline, as well as the task-specific BNs from Table 2.2 that yielded a performance drop of 5.76%. This observation corroborates the hypothesis from Sec. 2.4.6 that task-specific adaptation of the convolutions is essential for a model to perform competitively for all tasks. This is unlike the findings observed in multi-domain learning literature [BV17]. Additionally, we demonstrate that even without training entirely task-specific convolutions, as in Table 2.2, a performance boost can still be observed at a moderate increase in the number of parameters, seen in Fig. 2.4 when comparing *Task-specific Convs* and our reparameterized convolution depicted as *RCM*. As such, performance improvements from this baseline do not stem from an increase in network capacity. However, similar to [MRK19], RCM is a task-conditional multi-task model. This entails a separate forward pass for each task of interest, just like in the case of single-task models, but unlike the multi-task baseline that is often employed for latency constrained applications.

Response Initialization (RI) We investigate the effect on the performance when a more meaningful filter bank is utilized, RI (Sec. 2.3.3), against the filter bank learned by directly pre-training the RC architecture on ImageNet. The results are presented in Table 2.3. Compared to the RC model, the performance significantly improves from a 2.13% drop to a 1.12% drop. This observation clearly demonstrates that the filter bank

Table 2.4: Comparison with state-of-the-art methods on PASCAL-Context. The arrow indicates the direction for better performance. RCM outperforms the other methods, as indicated in bold.

Methods	Tasks					
	Edge ↑	SemSeg ↑	Parts ↑	Normals ↓	Sal ↑	Δ_m% ↓
Single-task	71.88	66.22	59.69	13.64	66.62	-
ASTMT [MRK19] (R-18 w/o SE)	71.20	64.31	57.79	15.06	66.59	3.49
ASTMT [MRK19] (R-26 w SE)	71.00	64.61	57.25	15.00	64.70	4.12
Series RA [RBV17]	70.62	65.99	55.32	14.27	66.08	2.97
Parallel RA [RBV18]	70.84	66.51	56.56	14.16	66.36	2.09
RCM (ours)	71.34	65.70	58.12	13.70	66.38	**0.99**

generated using our proposed RI approach is beneficial for better weight initialization.

Normalized Feature Fusion (NFF) We replace the unconstrained task-specific components of RC with the proposed NFF (Sec. 2.3.3). We demonstrate in Table 2.3 that NFF improves the performance no matter the initialization of the filter bank. Specifically, it improves the model using RI from 1.12% to 0.99%.

The architecture used for the remaining experiments utilized NFF and is initialized using the RI methodology. This architecture is denoted as RCM.

2.4.8 *Comparison to state-of-the-art*

In this work, we focus on comparing to task-conditional methods that can address MTL. We compare the performance of our method to Series Residual Adapter (RA) [RBV17] and Parallel RA[RBV18]. Series and Parallel RAs learn multiple visual domains by optimizing domain-specific residual adaptation modules on an ImageNet pre-trained backbone. Since both methods were developed for multi-domain settings, we optimize them using our own pipeline, ensuring a fair comparison amongst the methods while additionally benchmarking the capabilities of multi-domain methods in an MTL setup. We further report the performance of ASTMT [MRK19], which utilizes an architecture resembling that of Parallel RA [RBV18] with Squeeze-and-Excitation (SE) blocks [HSS18] and adversarial task disentanglement of gradients. Specifically, we report the performance of

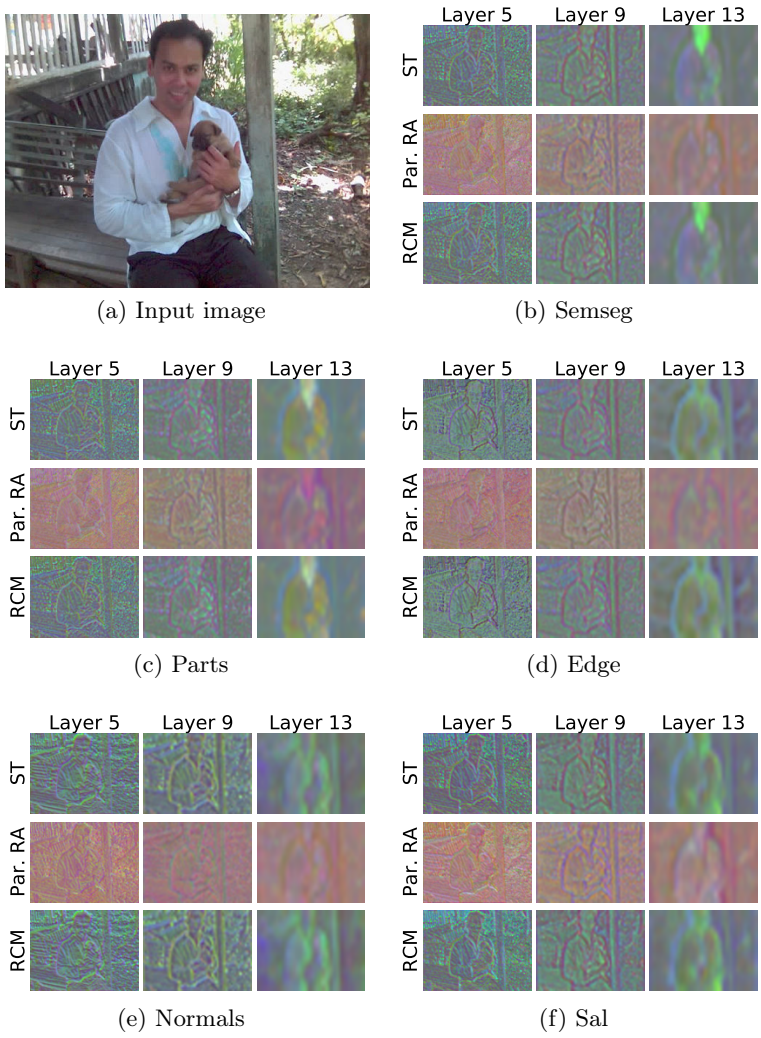
Table 2.5: Comparison with state-of-the-art methods on NYUD. The arrow indicates the direction for better performance. RCM outperforms the other methods, as indicated in bold.

Methods	Tasks				
	Edge ↑	SemSeg ↑	Normals ↓	Depth ↓	$\Delta_m\%$ ↓
Single-task	68.83	35.45	22.20	0.56	-
ASTMT [MRK19] (R-18 w/o SE)	68.60	30.69	23.94	0.60	6.96
ASTMT [MRK19] (R-26 w SE)	73.50	30.07	24.32	0.63	7.56
Series RA [RBV17]	67.56	31.87	23.35	0.60	5.88
Parallel RA [RBV18]	68.02	32.13	23.20	0.59	5.02
RCM (ours)	68.44	34.20	22.41	0.57	**1.48**

the models using a ResNet-26 (R-26) DeepLab-V3+ with SE as reported in [MRK19], and also optimize with the use of their codebase a ResNet-18 model without SE. The latter model uses an architecture resembling more closely that of the other methods since SE can be additionally incorporated in the others as well. We report the average performance drop with respect to our single-task baseline.

The results for PASCAL-Context (Table 2.4) and NYUD (Table 2.5) demonstrate that our method achieves the best performance, outperforming the other methods that make use of RA modules. This demonstrates that although the RA can perform competitively in multi-domain settings, placing the convolution in series without non-linearity is a more promising direction for the drastic adaptations required for different tasks in a MTL setup. Note that the task-conditional methods ASTMT (R-18 w/o SE) [MRK19], Series/Parllel RA [RBV17; RBV18], and RCM (ours) have nearly identical number of parameters. This spans from the primary difference being the location which the adaptation modules are located, and therefore the observed performance gains are not attributed to additional network capacity.

We visualize in Fig. 2.5 the learned representations of single-task, Parallel RA [RBV18], and RCM across tasks and network depths. For each task and layer combination, we compute a common PCA basis for the methods above and depict the first three principal components as RGB values in order to highlight similarities and differences between them. For all tasks and layers of the network, the representations of RCM closely resemble those of the single-task models. Simultaneously, Parallel RA is unable to adapt the convolution behavior to the extent required



(a) Input image (b) Semseg

(c) Parts (d) Edge

(e) Normals (f) Sal

Figure 2.5: We visualize the features of the input image (a) for the tasks of PASCAL-Context. The first row of each sub-figure corresponds to the responses of the single-task model (ST), the second row those of Parallel RA (Par. RA) [RBV18] and the final row of our proposed method (RCM). For all tasks and depths of the network, the responses of RCM closely resemble those of ST, in contrast to the responses of Par. RA. This is made apparent from the colours utilized by the different methods. The RGB values were identified from a common PCA basis across the three methods in order to highlight similarities and differences between them.

Table 2.6: Incremental learning experiments on PASCAL-Context using a network that originally has low-level task knowledge (Edge and Normals), in gray. The arrow indicates the direction for better performance. RCM outperforms the other methods, as indicated in bold.

Methods	Tasks					
	Edge ↑	Normals ↓	SemSeg ↑	Parts ↑	Sal ↑	$\Delta_m\%$ ↓
Single-task	71.88	13.64	66.22	59.69	66.62	-
ASTMT [MRK19] (R-18 w/o SE)	70.70	14.84	55.32	50.49	64.34	11.77
Series RA [RBV17]	70.62	14.27	65.99	55.32	66.08	2.83
Parallel RA [RBV18]	70.84	14.16	66.51	56.56	66.36	1.73
RCM (ours)	71.34	13.70	65.70	58.12	66.38	**1.26**

Table 2.7: Incremental learning experiments on PASCAL-Context using a network that originally has high-level task knowledge (SemSeg and Parts), in gray. The arrow indicates the direction for better performance. RCM outperforms the other methods, as indicated in bold.

Methods	Tasks					
	SemSeg ↑	Parts ↑	Edge ↑	Normals ↓	Sal ↑	$\Delta_m\%$ ↓
Single-task	66.22	59.69	71.88	13.64	66.62	-
ASTMT [MRK19] (R-18 w/o SE)	63.91	57.33	68.67	14.12	64.43	3.76
Series RA [RBV17]	65.99	55.32	70.62	14.27	66.08	2.39
Parallel RA [RBV18]	66.51	56.56	70.84	14.16	66.36	1.88
RCM (ours)	65.70	58.12	71.34	13.70	66.38	**0.52**

to be comparable to single-task models. This is made apparent from the colours utilized by the different methods.

2.4.9 *Incremental learning for multi-tasking*

We further evaluate the methods from Sec. 2.4.8 in the IL setup. In other words, we investigate the capabilities of the models to learn new tasks without the need to be completely retrained on the entire task dictionary. We divide the tasks of PASCAL-Context into three groups, **(i)** edge detection and surface normals (low-level tasks), **(ii)** saliency (mid-level task) and **(iii)** semantic segmentation and human parts segmentation (high-level tasks). IL experiments are conducted by allowing the base

network to initially use knowledge from either (i) or (iii), and report the capability for the optimized model to learn additional tasks without affecting the performance of the already learned tasks. We report the performance drop over the new tasks that were added later on. In the IL setup, ASTMT [MRK19] is initially trained using an R-18 backbone without SE (a comparable backbone to the competing methods for a fair comparison) on the subset of the tasks (either i or iii). New tasks can be incorporated by training their task-specific modules independently. On the other hand, Series RA, Parallel RA, and RCM, were designed to be inherently incremental due to directly optimizing only the task-specific modules. Consequently, their task-specific performance in the IL setup is identical to that reported in Sec. 2.4.8.

In Tables 2.6 and 2.7 we report the performance of tasks that are utilized to generate the initial knowledge of the model in grey, while in black the performance of the incrementally learned tasks. As shown in both tables, and in particular Table 2.6, ASTMT does not perform competitively in the IL experiments. This observation further demonstrates the importance of utilizing generic filter banks that can be adapted based on task-specific needs, in particular for IL setups. We consider research in generic multi-task filter banks to be a promising direction. Furthermore, RCM consistently outperforms the RA alternatives, attesting to the greater adaptation capabilities of filter banks when the adaptation takes place as a matrix multiplication rather than a residual operation.

2.5 CONCLUSION

We have presented a novel method of a convolutional operation reparameterization and its application to training multi-task learning architectures. These reparameterized architectures can be applied on a multitude of different tasks, and allow the CNN to be inherently incremental, while additionally eliminating task interference, all by construction. We evaluate our model on two datasets and multiple tasks, and show experimentally that it outperforms competing baselines that address similar challenges. We further demonstrate its efficacy when compared to the state-of-the-art task-conditional multi-task method, which is unable to tackle incremental learning.

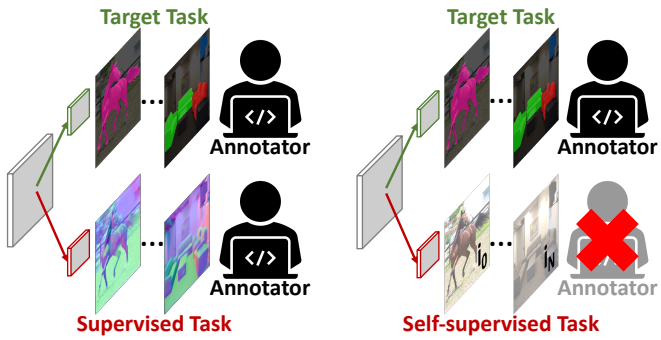
3

COMPOSITE LEARNING FOR DENSE PREDICTIONS

Multi-task learning promises better model generalization on a target task by jointly optimizing it with an auxiliary task. However, the current practice requires additional labeling efforts for the auxiliary task, while not guaranteeing better model performance. In this chapter, we find that jointly training a dense prediction (target) task with a self-supervised (auxiliary) task can consistently improve the performance of the target task, while eliminating the need for labeling auxiliary tasks. We refer to this joint training as Composite Learning (CompL). Experiments of CompL on monocular depth estimation, semantic segmentation, and boundary detection show consistent performance improvements in fully and partially labeled datasets. Further analysis on depth estimation reveals that joint training with self-supervision outperforms most labeled auxiliary tasks. We also find that CompL can improve model robustness when the models are evaluated in new domains. These results demonstrate the benefits of self-supervision as an auxiliary task, and establish the design of novel task-specific self-supervised methods as a new axis of investigation for future multi-task learning research.

3.1 INTRODUCTION

Learning robust and generalizable feature representations have enabled the utilization of CNNs on a wide range of tasks. This includes tasks that require efficient learning due to limited annotations. A commonly used paradigm to improve the generalization of target tasks is MTL, the joint optimization of multiple tasks. MTL exploits domain information contained in the training signals of related tasks as an inductive bias in the learning process of the target task [Car97; Car98]. The goal is to find joint representations that better explain the optimized tasks. MTL has demonstrated success in tasks such as instance segmentation [DHS16] and depth estimation [Che+19], amongst others. In reality, however, such performance improvements are not common when naively selecting the jointly optimized tasks [Kok17]. To complicate things further, the



(a) Multi-Task Learning (b) Composite Learning (ours)

Figure 3.1: The generalization of target tasks can be improved by jointly optimizing with a related auxiliary task. (a) In traditional multi-task learning, one uses labeled auxiliary tasks that require manual annotation efforts. (b) In this chapter, we show that jointly training a dense task with a self-supervised task can consistently improve the performance, while eliminating the need for additional labeling efforts.

relationship between tasks for MTL is also dependent on the learning setup, such as training set size and network capacity [Sta+20]. As a consequence, MTL practitioners are forced to iterate through various candidate task combinations in search of a synergetic setting. This empirical process is arduous and expensive since annotations are required *a priori* for each candidate task.

In this chapter, we find that the joint optimization of a dense prediction (target) task with a self-supervised (auxiliary) task improves the performance on the target task, outperforming traditional MTL practices. We refer to this joint training as Composite Learning (CompL), inspired by material science where two materials are merged to form a new one with enhanced properties. The benefits and intuition of CompL resemble those of traditional MTL, however, CompL exploits the label-free supervision of self-supervised methods. This facilitates faster iterations through different task combinations, and eliminates manual labeling effort for auxiliary tasks from the process.

We provide thorough evaluations of CompL on three dense prediction target tasks with different model structures, combined with three self-supervised auxiliary tasks. The target tasks include depth estimation, semantic segmentation, and boundary detection, while self-supervised tasks include rotations [GSK18], MoCo [He+20; Che+20b],

and DenseCL [Wan+21d]. We find that jointly optimizing with self-supervised auxiliary tasks consistently outperforms ImageNet-pretrained baselines. The benefits of CompL are most pronounced in low-data regimes, where the importance of inductive biases increases [Bax00]. We also find that jointly optimizing monocular depth estimation with a self-supervised objective can outperform most labeled auxiliary tasks. CompL can additionally improve semantic segmentation and boundary detection model robustness, when evaluated on new domains. Our experiments demonstrate the promise of self-supervision as an auxiliary task. We envision these findings will establish the design of novel task-specific self-supervised methods as a new axis of investigation for future multi-task learning research.

3.2 RELATED WORK

Multi-Task Learning (MTL) MTL aims to enhance performance and robustness of a predictor by jointly optimizing a shared representation between several tasks [Car97]. This is accomplished by exploiting the domain-specific information contained in the training signal of one task (e.g., semantic segmentation), to more informatively select hypotheses for other tasks (e.g., depth), and vice versa [RPC17; Bru+21]. For example, pixels of class "*sky*" will always have a larger depth that those of class "*car*" [Sah+21]. If non-related tasks are combined, however, the overall performance degrades. This is referred to as task-interference and has been well documented in the literature [MRK19; Kan+20]. However, no measurement of task relations can tell us whether performance gain can be achieved without training the final models. Although several works have shown that while MTL can improve performance, it requires an exhaustive manual search of task interactions [Sta+20], and labeled datasets with many tasks. In this chapter we also jointly optimize a network on multiple tasks, but we instead evaluate the efficacy of self-supervision as an auxiliary task. This enables the use of joint training in any dataset and eliminates expensive annotation efforts that do not guarantee performance gains. To further improve performance of a target task, [Hoy+21; Gui+20b; BKK22; Geo+21] designed specialised architectures for a predefined set of tasks. These architectures do not generalize to other tasks. On the other end, Liu et al. [LDJ19] aim to learn a sub-class labelling problem as an auxiliary task, i.e. for class dog learn the breed subclass, however the notion of subclass does not generalize to dense tasks like depth estimation. Instead, we conduct

a systematic investigation using a common pipeline, applicable to any dense target task. This enables the easy switching of different supervised target tasks or auxiliary self-supervised tasks, without requiring any architectural changes, enabling the wider reach of joint training across tasks and datasets.

Transfer learning Given a large labeled dataset, neural networks can optimize for any task, whether image-level [KSH12], or dense [HGD19]. In practice, however, large datasets can be prohibitively expensive to acquire, giving rise to the transfer learning paradigm. The most prominent example of transfer learning is the fine-tuning of an ImageNet [Den+09] pre-trained model on target tasks such as semantic segmentation [LSD15], or monocular depth estimation [Fu+18]. However, ImageNet models do not always provide the best representations for all downstream tasks, raising interest in finding task relationships for better transfer capabilities [Zam+18]. In this chapter we are not interested in learning better pre-trained networks for knowledge transfer. Rather, we start from strong transfer learning baselines and improve generalization by jointly optimizing the target and auxiliary tasks.

Self-supervised learning Learning representations that can effectively transfer to downstream tasks, coupled with the cost associated with the acquisition of large labeled datasets, has given rise to self-supervised methods. These methods can learn representations through explicit supervision on pre-text tasks [DGE15; GSK18], or through contrastive methods [Che+20a; He+20]. Commonly, self-supervised methods aim to optimize a given architecture, yielding better pre-training models for fine-tuning on the target task [DGE15; GSK18; Che+20a; He+20; Ghi+21; Wan+21d; ND20; Li+22]. We instead utilized such pre-trained models as a starting point and fine-tune on both the target and self-superivsed auxiliary tasks jointly, rather than just the target task, to further improve performance and robustness. More recently, supervised tasks have been used in conjunction with self-supervised techniques by exploiting the labels to guide contrastive learning. This can be seen as a form of sampling guidance and has been utilized in classification [Kho+20], semantic segmentation [Wan+21b], and tracking [Pan+21]. These methods differ from our work as they require target task labels to optimize the self-supervised objective, while our self-supervised objectives are independent of the target labels and can be applied on any set of images. Instead, Deng et al. [DGZ21] jointly train a model for classification and rotation, but utilize the rotation performance at test time as a proxy to the classification performance. The goal of this work is instead to improve the target

task's performance and robustness. More closely to our work, Gidaris et al. [Gid+19] and Zhai et al. [Zha+19a] jointly train classification and self-supervised objectives under a semi-supervised training protocol. We also perform joint training with a self-supervised task, however, we follow a more general MTL methodology, and investigate whether self-supervised tasks can provide inductive bias to dense tasks.

Robustness Robust predictors are important to ensure their performance under various conditions during deployment. Recent works have focused on improving different aspects of robustness, such as image corruption [HD19], adversarial samples [Zha+19b], and domain shifts [Yu+20]. More related to our work, Hendrycks et al. [Hen+19] jointly train classification and self-supervised rotation, demonstrating that the strong regularization of the rotations improves model robustness to adversarial examples, and label or input corruptions. Wang et al. [Wan+21c] similarly used joint training but employed both image and video-level self-supervised tasks and found them to improve the model's robustness to domain shifts for object detection. We also evaluate the effect of joint training on robustness to unseen datasets, but focus on dense prediction tasks.

3.3 COMPOSITE LEARNING

In this section, we introduce and motivate Composite Learning (CompL). Specifically, Sec. 3.3.1 formalizes the problem setting, Sec. 3.3.2 describes the self-supervised methods investigated, and Sec. 3.3.3 lists the network structure choices in our study.

3.3.1 *Joint Learning with Supervised and Self-Supervised Tasks*

Multi-task learning may improve the model robustness and generalizability. We aim to investigate the efficacy of joint training with self-supervision on dense prediction tasks as the targets. The shared representation between the target task t and an auxiliary task a may be more effective than training on t alone.

In the traditional MTL setup, the label sets Y_t, and Y_a, are manually labeled. In contrast, the auxiliary labels Y_a in CompL are implicitly created in the self-supervised task. Formally, CompL aims to produce the two predictive functions $f_t(\theta_s, \theta_t) : \mathcal{X}_t \to \mathcal{Y}_t$ and $f_a(\theta_s, \theta_a) : \mathcal{X}_a \to \mathcal{Y}_a$ that map the input space $\mathcal{X}$ to the label space $\mathcal{Y}$, where f_t and f_a share parameters θ_s and have disjoint parameters $\theta_{\{t,a\}}$. During inference we

are only interested in f_t, however, we hypothesize that we can learn a more effective parameterization through the above weight sharing scheme. In our investigation, f_t and f_a are trained jointly using samples (X_t, y_t) and (X_a, y_a), where X represents input images and y the corresponding labels.

The overall optimization objective therefore becomes

$$\min_{\theta_s,\theta_t,\theta_a} \mathcal{L}^t((X_t, y_t); \theta_s, \theta_t) + \lambda \mathcal{L}^a((X_a, y_a); \theta_s, \theta_a), \tag{3.1}$$

where $\mathcal{L}^t$ and $\mathcal{L}^a$ are the losses for the supervised and self-supervised tasks respectively, and λ is a scaling factor controlling the magnitude and importance of the self-supervised task.

The experiments in this chapter use the same dataset for both the target and auxiliary tasks. We additionally train our models using different-sized subsets (X'_t, y'_t) for the target task, where $X'_t \subseteq X_t = X_a$. However, the above is not a necessary condition for CompL, meaning the self-supervised task could be trained on an independent dataset.

Training method We jointly optimize two objectives. We construct a minibatch by sampling at random independently from the two training sets. For simplicity, we sample an identical number of images from each training set. The input images X_t and X_a are treated independently. This enables us to apply task/method-specific augmentations to each task input without causing task conflicts. We apply the baseline augmentations to X_t, ensuring a fair comparison with our single-task baselines. X_a used for self-supervised training is instead processed with the proposed task-specific augmentations for each method investigated. These augmentations include Gaussian blur and rotation. They can significantly degrade performance for dense tasks if applied on the target task, but they are important for self-supervision. Therefore, by using distinct augmentations on two tasks, we can minimize performance degradation brought by training the auxiliary tasks.

3.3.2 *Self-Supervised Methods in Our Study*

Rotation (Rot) Gidaris et al. [GSK18] proposed to utilize 2-dimensional rotations on the input images to learn feature representations. Specifically, they optimize a classification model to predict the rotation angles, equally spaced in $[0°, 360°)$. Joint optimization with self-supervised rotation has demonstrated success in semi-supervised image classification [Gid+19; Zha+19a], and enhanced robustness to input/output cor-

ruptions [Hen+19], making it a prime candidate for further investigation in a dense prediction setting.

Global contrastive Global contrastive methods treat every image as its own class, while artificially creating novel instances of said class through random data augmentations. In this work, we evaluate contrastive methods using Momentum Contrast (MoCo) [He+20], and specifically MoCo v2 [Che+20b]. These methods formulate contrastive learning as dictionary look-up, enabling for the construction of a large and consistent dictionary of size $|Z|$ without the need for large batch sizes, a common challenge amongst dense prediction tasks [Che+18a]. MoCo is optimized using InfoNCE [OLV18], a contrastive loss function defined as

$$\mathcal{L} = -\log \frac{\exp\left(z^{+}/\tau\right)}{\sum_{z \in Z} \exp\left(z/\tau\right)}. \tag{3.2}$$

InfoNCE is a softmax-based classifier that optimizes for distinguishing the positive representation z^{+} from the $|Z| - 1$ negative representations. The temperature τ is used to control the smoothness of the probability distribution, with higher values resulting in softer distributions.

Local contrastive In dense predictions tasks, we desire a fine-grained pixel wise prediction rather than a global one. As such, we further investigate the difference between global contrastive MoCo v2 [Che+20b], and its variant DenseCL [Wan+21d], that includes an additional contrastive loss acting on local representations.

3.3.3 *Network Structures*

Dense prediction networks are initially pre-trained on classification, and then modified according to the downstream task of interest, e.g., by introducing dilations [YK16b]. In our investigation, we jointly optimize heterogeneous tasks such as a dense prediction task and image rotations. Therefore, our networks call for special structure considerations. This section presents the details.

Dense prediction networks Common dense prediction networks use an encoder-decoder structure [RFB15; BKC17], maintain a constant resolution past a certain network depth [YKF17], or even utilize both high and low representation resolutions in multiple layers of the network [Wan+20a]. Due to the large differences among networks, we opt to treat the entire network as a single unit, and only utilize the last feature

representation of the networks for the task-specific predictions. In other words, we branch out at the last layer and employ a single task-specific module for the predictions. This ensures that our findings do not depend on network structures, and it is easy to generalize to new network designs.

We perform our experiments on DeepLabv3+ [Che+18a] based on ResNets [He+16]. The networks demonstrated competitive performance on a large number of dense prediction tasks, such as semantic segmentation, and depth estimation and has been used extensively when jointly learning multiple tasks [MRK19; Bru+20]. Our investigation is primarily on the smaller ResNet-26 architecture for easy comparison with existing MTL results. As it is a common practice in dense prediction tasks, we initialize the ResNet encoder with ImageNet pre-trained weights, unless stated otherwise.

Task-specific heads The final representation of the dense prediction networks is utilized in two task-specific modules. The first module, consisting of a 1×1 convolutional layer, generates the predictions of the supervised task, with the output dimension being task dependent, such as the number of classes. The second prediction head is specific for self-supervised tasks. Unlike the supervised prediction head, the self-supervised prediction head is utilized only during network optimization, and is discarded at test time. The features for Rot and MoCo are first pooled with a global average pooling layer. Rot is then processed by a fully connected layer with output dimensions equal to 4, number of potential rotations, while MoCo is processed by 2-layer MLP head with output dimensions equal to 128, feature embedding dimension. DenseCL, on the other hand, generates two outputs. The first one is identical to MoCo for the global representation, while for the second representation, the initial dense features are pooled to a smaller grid size, and then processed with two 1×1 convolutional layer to get the local feature representations.

Normalization Large CNNs are often challenging to train, and thus utilize Batch Normalization (BN) to accelerate training [IS15]. In self-supervised training, BNs often degrade performance due to intra-batch knowledge transfer among samples. Workarounds include shuffling BNs [He+20; Che+20b], using significantly larger batch sizes [Che+20a], or even replacing BNs altogether [Hen20]. To ensure BNs will not affect our study, and findings can be attributed to the jointly trained tasks, we replace BNs with Group Normalization (GN) [WH18]. We chose GN as it yielded the best performance when trained on ImageNet [WH18]. However, other normalization layers that are not affected by batch statistics can also be utilized, such as layer [BKH16] and instance [UVL16] normalizations.

3.4 EXPERIMENTS

In this section we investigate the effects of jointly training dense predictions and self-supervised tasks. To systematically assess the effect of joint learning in label-deficient cases, we use different-sized subsets (X'_T, y'_t) of the full target task data (X_T, y_t), i.e., $(X'_T, y'_t) \subseteq (X_T, y_t)$. To ensure consistent contribution from the auxiliary task, we use the full data split (X_A, y_a) for the self-supervised task, unless stated otherwise.

3.4.1 *Implementation details*

Codebase details We base our experiments on the VIsion library for state-of-the-art Self-Supervised Learning (VISSL) [Goy+21], released under the MIT License. VISSL includes implementations of self-supervised methods, and was adapted to enable for the joint optimization of the existing algorithms with supervised methods, such as semantic segmentation. All experiments were conducted in our internal cluster using single V-100 GPUs. Due to the considerable costs associated with multiple runs, we run all experiments with a random seed of 1, the default setting of VISSL.

Generic hyperparameters We sample 8 images at random from each of the target and auxiliary training sets. We apply the baseline augmentations to target samples, namely, random horizontal flipping, random image scaling in the range [0.5, 2.0] in 0.25 increments, and then crop or pad the image to ensure a consistent size. The auxiliary loss is scaled by λ. We found 0.2 works best for MoCo and DenseCL, and 0.05 for Rot. The model is optimized using stochastic gradient decent with momentum 0.9, weight decay 0.0001, and the "poly" learning rate schedule [Che+17].

Hyperparameter λ During training, the auxiliary loss is scaled by the hyperparameter λ, weighting the contribution of the auxiliary self-supervised task. The hyperparameter λ was selected by performing a logarithmic grid search, as commonly done in MTL literature, chosen from the set $\{0.05, 0.1, 0.2, 0.5, 1.0\}$. We found the performance of the models to be consistent when λ is in the range of 0.1 to 0.5, as seen in Table 3.1. The performance quickly degrades for values an order of magnitude larger as the model prioritizes the auxiliary task over the target task, while smaller values converge to the baseline performance.

Table 3.1: Ablation of the λ parameter for the semantic segmentation model trained jointly with DenseCL. The model yields comparable performance for all three values.

λ	Labeled Data	
	10%	50%
0.1	57.21	68.64
0.2	**57.33**	**68.81**
0.5	57.27	68.79

Memory bank MoCo [He+20] and DenceCL [Wan+21d] utilized a memory bank to enlarge the number of negative samples observed during training, while keeping a tractable batch size. Specifically, both methods use a memory bank of size 65,536. All the datasets we used in our study are of a smaller magnitude compared to that memory bank, e.g. 10,582 and 795 for PASCAL VOC 2012 (aug.) [Har+11] and NYUD-v2 [Sil+12], respectively. We therefore set the memory bank to have the same size as the training dataset, yielding a single positive per sample, and therefore allowing for the direct use of the InfoNCE loss [OLV18]. A larger memory bank can also be used, however the contrastive loss would need to be adapted to account for multiple positives [Kho+20].

Image cropping We use nearly identical augmentations to those proposed in MoCo v2 [Che+20b] for the self-supervised methods of [He+20; Wan+21d], but found it beneficial to modify image cropping. In most classification datasets, each image is comprised of a single object, and thus low overlapping crops can still include the same object. In dense tasks such as semantic segmentation, low overlapping crops can contain different objects (Fig. 3.2). We follow the practice of [Tia+20] and find a constant crop size and distance between the two patches for each task. We empirically find that square crops of size 384 with a distance of 32 pixels on both axis works best for semantic segmentation, crops of size 283×373 (to maintain input size ratio) with a distance of 8 pixels worked best for depth, and square crops of size 320 with a distance of 4 pixels worked best for boundary estimation.

DenceCL global vs local contrastive DenseCL, as discussed in Sec. 3.3.2, includes a global and a local contrastive term. The importance of the local contrastive term is weighed by a constant parameter. The original paper found that 0.7 for local contrastive and 0.3 for global contrastive performed best for detection, but used 0.5 to strike a balance between the downstream performance on detection and classification.

(a) Input image

(b) Blue crop

(c) Purple crop

Figure 3.2: Low overlapping crops can be semantically different. This is more apparent in dense prediction datasets where multiple objects can be present in each image.

In our study, we also found 0.7 for local contrastive yields the best performance, and as such, used it for all DenseCL experiments.

3.4.2 *Monocular Depth Estimation*

We first evaluate CompL on monocular depth estimation. Monocular depth estimation is a widely used dense prediction task, and is typically casted as a regression problem.

Experimental protocol Monocular depth estimation is explored on NYUD-v2 [Sil+12], comprised of 795 train and 654 test images from indoor scenes, and evaluated using the root mean squared error (RMSE) metric. All models are trained for 20k iterations, corresponding to 200 epochs of the fully labeled dataset, with an input image size of 425×560, and are optimized with the $\mathcal{L}_1$ loss.

Joint optimization Table 3.2 presents the performance of the single-task baseline, "Depth", and the models trained jointly with different self-supervised tasks, "Depth + *Task name*". We find that joint training

Table 3.2: Monocular depth estimation performance in RMSE on NYUD-v2. '→' denote transfer learning methods, while '+' denote joint training (CompL). Initialization with DenseCL coupled with DenseCL joint training outperforms all other methods.

Model	Labeled Data				
	5%	10%	20%	50%	100%
Depth	0.8871	0.8120	0.7471	0.6655	0.6223
Rot → Depth	1.0830	1.0120	0.9114	0.8322	0.7822
MoCo → Depth	0.8758	0.7708	0.7113	0.6311	0.5890
DenseCL → Depth	0.8736	0.7726	0.7152	0.6321	0.5982
Depth + Rot	0.8762	0.8071	0.7298	0.6460	0.6107
Depth + MoCo	0.8501	0.7955	0.7206	0.6434	0.6000
Depth + DenseCL	0.8479	0.7866	0.7131	0.6420	0.5990
MoCo → MoCo + Depth	0.8614	0.7732	0.7008	0.6220	0.5773
DenseCL → DenseCL + Depth	**0.8468**	**0.7641**	**0.6989**	**0.6157**	**0.5690**

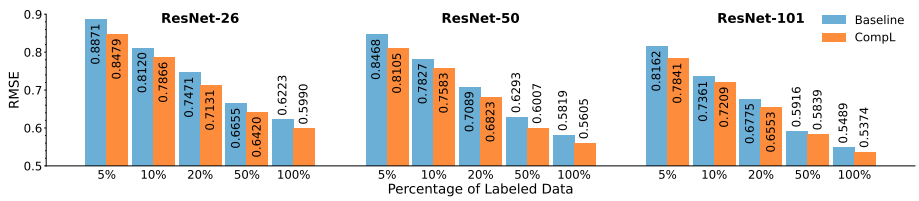
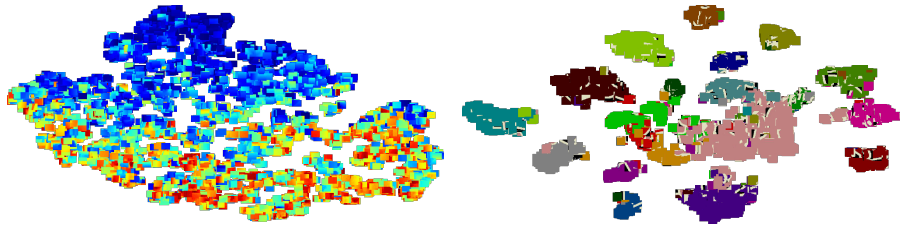


Figure 3.3: Monocular depth estimation performance in RMSE on different ResNet encoders. Use of CompL (orange) denotes the addition of the best performing self-supervised objective (DenseCL). CompL consistently outperforms the baselines in all experiments.

with any self-supervised task consistently improves the performance of the target task, even in the fully labeled dataset. In particular, joint training with self-supervision yields the biggest performance improvements on the lower labeled percentages, where the importance of inductive bias increases [Bax00]. These findings are consistent also when utilizing stronger ResNet encoders, as depicted in Fig. 3.3 for the best performing self-supervised DenseCL method.

DenseCL contrasts both local and global representations, yielding more effective representations for dense task pre-training, as compared to the image-level self-supervised tasks. We find this to also be the case in our joint-training setup, where local representations help guide the optimization of depth. To better understand the benefit of utilizing DenseCL for joint training with depth, we visualize the representations



(a) Monocular depth estimation. (b) Semantic segmentation.

Figure 3.4: t-SNE visualization of the DenseCL local representations. The representations are depicted using their ground-truth maps. Specifically, (a) depth values for monocular depth estimation and (b) semantic patches for semantic segmentation. The local representations adapt to the target task, i.e., (a) smooth depth variation for the regression task while (b) clusters are formed for the classification task.

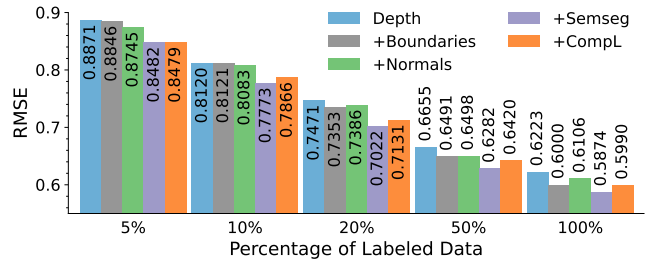


Figure 3.5: Monocular depth estimation performance in RMSE on NYUD-v2 when trained with additional auxiliary tasks. CompL can improve depth more than training with boundary or normal predictions. Semantic segmentation can improve the depth prediction more, but it requires expensive manual annotations.

in Fig. 3.4a using a t-SNE plot [VH08]. Specifically, we depict the latent representations of DenseCL using their corresponding ground-truth depth measurements. The depth values smoothly transition from larger distances (in red) to smaller distances (in blue). This indicates that the DenseCL objective, which is discriminative by construction, promotes a smooth variation in the representations when combined with a regression target objective.

Traditional MTL In order to determine how CompL compares to traditional MTL, we evaluate and compare the effect of using labeled auxiliary tasks. Specifically, we investigate the effect of the remaining three tasks of NYUD-v2, that is, boundaries, normals, and semantic segmentation, in Fig. 3.5. For fair comparisons to CompL, the auxiliary

Table 3.3: Monocular depth estimation performance in RMSE on NYUD-v2. Both supervised and self-supervised objectives use identical dataset subsets. CompL denotes the addition of the best performing self-supervised objective, DenseCL, and yields consistent improvements.

CompL	Dataset Size				
	5%	10%	20%	50%	100%
	0.8871	0.8120	0.7471	0.6655	0.6223
✓	**0.8840**	**0.8080**	**0.7305**	**0.6508**	**0.5990**

tasks also use the entire dataset. CompL consistently outperforms the use of labeled boundaries and normals as auxiliary tasks. This is particularly pronounced in the lower data splits where the contribution of CompL becomes more prominent, while boundaries and normals contribute less. Surface normals, derivatives of depth maps, could be expected to boost depth prediction due to their close relationship. However, we find it to help only marginally. On the other hand, joint training with semantic segmentation consistently improves the baseline performance, which aligns with findings in the previous works [Che+19; Gui+20a; Jia+18]. These results exemplify the importance of an arduous iteration process in search of a synergistic auxiliary task, where knowledge of label interactions are not necessarily helpful. This process is further complicated when additional auxiliary task annotations are needed. Therefore, eliminating manual labeling from auxiliary tasks opens up a new axis of investigation for the future of multi-task learning research as it can enable faster iterations in task interaction research.

Transfer learning The experiments have so far shown that joint training with self-supervision can enhance performance, and in most cases outperforms traditional MTL practices. Notably, outperforming the baselines even when all models are initialized with ImageNet pre-trained weights, a strong transfer learning baseline. However, is ImageNet pre-training the best initialization for Depth, and how does it compare to self-supervised pre-training? In Table 3.2 we repeat the baseline experiments starting from self-supervised pre-training, ("*Initial task* → Depth"). In depth estimation, the contrastive methods gain the advantage and outperform the joint training methods. However, our proposed method is not limited by the initialization used. We find that initialization with MoCo or DenseCL weights coupled with joint training ("*Initial task* → *Initial task* + Depth") can increase the performance even further, giving the best performing models.

Joint optimization on identical dataset subsets Table 3.3 presents the performance of the monocular depth estimation single-task baseline and the best performing self-supervised task, DenseCL. Unlike Table 3.2, however, Table 3.3 reports the performance when both supervised and self-supervised objectives use the same subset for optimization. Consistent improvements across all dataset splits are still observed.

3.4.3 *Semantic Segmentation*

We additionally evaluate semantic segmentation. Semantic segmentation is representative for discrete labeling dense predictions.

Experimental protocol Semantic segmentation (Semseg) experiments are conducted on PASCAL VOC 2012 [Eve+10], and specifically the augmented version (aug.) from [Har+11], that provides 10,582 train and 1,449 test images. We evaluate performance in terms of mean Intersection-over-Union (mIoU) across the classes. All models are trained for 80k iterations, accounting for 60 epochs of the fully labeled dataset, and are optimized with the cross-entropy loss with image input size of 512×512.

Joint optimization Table 3.4 present the performance of the single-task baseline and the models trained jointly with different self-supervised tasks. In contrast to findings from classification literature [Gid+19; Zha+19a], joint training with Rot minimally affects the performance in most cases, with lower labeled percentages even incurring a performance degradation. On the other hand, the contrastive methods increase performance on all labeled splits, with lower labeled percentages incurring the biggest performance improvement. These findings are once again consistent when utilizing stronger ResNet encoders, as depicted in Fig. 3.6 for the best performing self-supervised method DenseCL. Similar to depth, we further visualize in Fig. 3.4b the latent representations contrasted by DenseCL, and depict them with their ground-truth semantic maps. Unlike in depth regression, where the representations were smooth due to the continuous nature of the problem, the DenseCL representations for semantic segmentation form clusters given the discriminative nature of semantic segmentation.

Robustness to zero-shot dataset transfer So far we have only evaluated on the same distribution as that used for training, however, distribution shifts during deployment are common. We therefore investigate the generalization capabilities to new and unseen datasets. We evaluate the zero-shot capabilities of the models on the challenging

Table 3.4: Semantic segmentation performance in mIoU on the PASCAL VOC dataset. '→' denote transfer learning methods, while '+' denote joint training (CompL). Joint training with DenseCL significantly outperforms the "Semseg" baselines.

Model	Labeled Data						
	1%	2%	5%	10%	20%	50%	100%
Semseg	30.82	37.66	49.95	55.17	61.30	67.38	70.42
Rot → Semseg	10.35	12.43	18.29	24.71	29.21	35.43	39.46
MoCo → Semseg	31.55	37.55	48.60	53.27	58.74	64.04	68.09
DenseCL → Semseg	34.89	39.72	50.96	55.60	61.13	65.71	69.56
Semseg + Rot	28.75	36.81	50.46	56.21	62.17	67.96	70.52
Semseg + MoCo	32.90	40.31	52.18	56.50	62.49	68.40	71.15
Semseg + DenseCL	33.51	40.91	52.76	**57.33**	**63.22**	**68.81**	**71.16**
DenseCL → Semseg + DenseCL	**36.32**	**41.24**	**52.94**	56.87	62.71	65.89	69.81

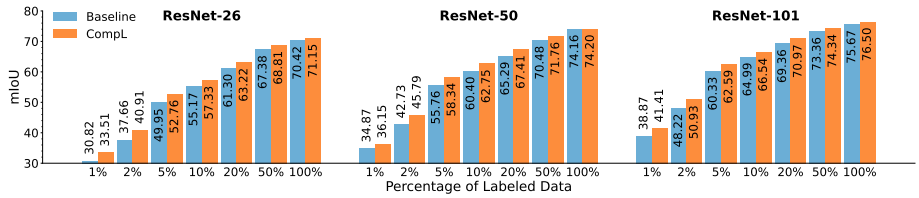


Figure 3.6: Semantic segmentation performance in mIoU on different ResNet encoders. Use of CompL (orange) denotes the addition of the best performing self-supervised objective (DenseCL). CompL consistently outperforms the baselines in all experiments.

BDD100K [Yu+20], a diverse driving dataset, in Fig. 3.7 and Table 3.5. The test frames from BDD100K are therefore significantly different to those observed during training, making zero-shot transfer particularly interesting due to the large domain shift. We report the mIoU with respect to the shared classes between the two datasets. Please refer to the supplementary for the table of the BDD100K experiments.

We find that Rot often performs worse than the baseline model. This yields dissimilar findings to classification [Hen+19] that observed increased robustness, attributed to the strong regularization induced by the joint training. For Semseg, such regularizations degrade the fine-grained precision required. Joint training with DenseCL significantly outperforms all other self-supervised methods. While MoCo was comparable to DenseCL on VOC (Table 3.4), we find that contrasting on the local level plays a big role in improving robustness. Interestingly, when using 100% of the

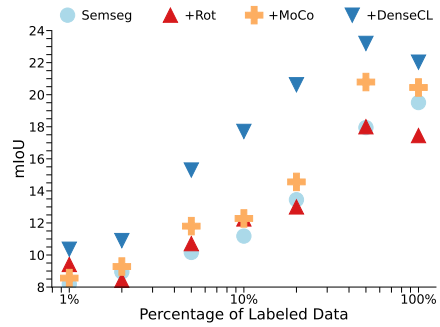


Figure 3.7: Semantic segmentation performance in mIoU trained on PASCAL VOC and evaluated on BDD100K. The local contrastive loss of DenseCL provides significant robustness improvements.

Table 3.5: Performance of semantic segmentation in mIoU trained on PASCAL VOC and evaluated on BDD100K. The local contrastive loss of DenseCL provides significant robustness improvements.

Model	Labeled Data						
	1%	2%	5%	10%	20%	50%	100%
Semseg	8.18	8.95	10.16	11.18	13.45	17.95	19.51
Semseg + Rot	9.41	8.42	10.71	12.25	13.00	18.00	17.45
Semseg + MoCo	8.56	9.28	11.8	12.28	14.56	20.79	20.45
Semseg + DenseCL	**10.36**	**10.90**	**15.30**	**17.71**	**20.62**	**23.20**	**22.03**

data points, performance on all methods utilizing self-supervision is lower than when using 50% of the labels. We conjecture that, using the fully labeled split decreases the influence of self-supervision, making the model more prone to overfit to the training dataset and lose generalizability.

We additionally evaluate how the models trained on PASCAL VOC from Table 3.4 ("Semseg" and "Semseg + *Task name*") perform without re-training on COCO [Lin+14] on the same classes. As seen in Table 3.6 and Fig. 3.8, joint training with the contrastive methods consistently outperform across all percentage splits, with the lower labeled percentages observing the biggest improvement.

Transfer learning Table 3.4 additionally reports the baseline experiments starting from self-supervised pre-training (indicated by "*Initial task* → Semseg"), or additionally optimized with the best performing DenseCL method, as in the Depth experiments. Joint training with self-supervision consistently outperforms the sequential training counterpart, and in the majority of the cases by a significant margin. In other words,

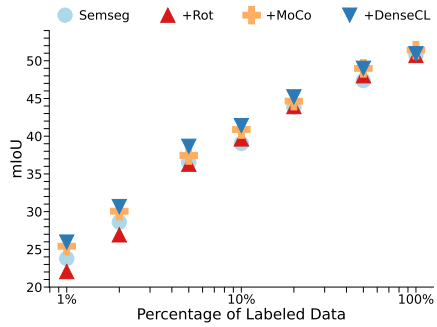


Figure 3.8: Performance of semantic segmentation in mIoU trained on PASCAL VOC and evaluated on COCO. The local contrastive loss of DenseCL provides consistent robustness improvements.

Table 3.6: Performance of semantic segmentation in mIoU trained on PASCAL VOC and evaluated on COCO. The local contrastive loss of DenseCL provides consistent robustness improvements.

Model	Labeled Data						
	1%	2%	5%	10%	20%	50%	100%
Semseg	23.78	28.62	36.53	39.05	43.85	47.37	50.76
Semseg + Rot	22.05	26.92	36.29	39.64	43.93	48.01	50.70
Semseg + MoCo	25.42	30.06	37.44	40.88	44.64	48.99	**51.41**
Semseg + DenseCL	**25.96**	**30.67**	**38.66**	**41.40**	**45.21**	**49.00**	50.93

CompL consistently reports performance gains when initializing with either ImageNet or DenseCL.

Joint optimization on identical dataset subsets Table 3.7 presents the performance of the semantic segmentation single-task baseline and the best performing self-supervised task DenseCL. Similar to Table 3.3, both objectives use the same subset for optimization. Consistent improvements across all dataset splits are again observed.

3.4.4 *Boundary Detection*

Boundary detection is another common dense prediction task. Unlike depth prediction and semantic segmentation, the target boundary pixels only account for a small percentage of the overall pixels. We find that CompL significantly improves the model robustness for boundary detection.

Table 3.7: Semantic segmentation performance in mIoU on PASCAL VOC. Both supervised and self-supervised objectives use identical splits. CompL denotes the addition of the best performing self-supervised objective, DenseCL, and yields consistent improvements.

CompL	Dataset Size						
	1%	2%	5%	10%	20%	50%	100%
	30.82	37.66	49.95	55.17	61.30	67.38	70.42
✓	**31.59**	**38.85**	**50.87**	**56.45**	**61.92**	**68.06**	**71.15**

Table 3.8: Boundary detection performance in ODS F-score on the BSDS500 dataset. '→' denote transfer learning methods, while '+' denotes joint training. Performance improvements are marginal, in contrast to the findings for other target tasks.

Model	Labeled Data			
	10%	20%	50%	100%
Boundaries	71.10	73.50	75.90	76.80
Rot → Boundaries	60.20	62.80	66.00	67.70
MoCo → Boundaries	71.00	73.40	75.60	76.40
DenseCL → Boundaries	68.90	71.70	75.40	75.90
Boundaries + Semseg	70.60	73.30	75.60	**76.90**
Boundaries + Rot	69.70	73.00	75.70	76.60
Boundaries + MoCo	**71.30**	73.80	**76.20**	**76.90**
Boundaries + DenseCL	**71.30**	**73.90**	76.00	76.20

Experimental protocol We study boundary detection on the BSDS500 [Arb+10] dataset, consisting of 300 train and 200 test images. Since the ground truth labels of BSDS500 are provided by multiple annotators, we follow the approach of [XT15] and only count a pixel as positive if it was annotated as positive by at least three annotators. Performance is evaluated using the Optimal-Dataset-Scale F-measure (ODS F-score) [MFM04]. All models are trained for 10k iterations on input images of size 481×481. Following [XT15], we use a cross-entropy loss with a weight of 0.95 for the positive and 0.05 for the negative pixels.

Joint optimization Table 3.8 presents the performance of the single-task baseline and the models trained jointly with different self-supervised tasks. Compared to the previous two tasks, boundary detection is marginally improved by CompL. Since convolutional networks are biased towards recognising texture rather than shape [Gei+18], we hypothesize that the

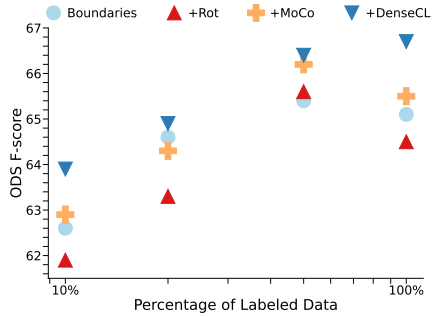


Figure 3.9: Boundary detection performance in ODS F-score trained on BSDS and evaluated on NYUD. The additional local contrast of DenseCL increases robustness to zero-shot dataset transfer.

supervisory signal of contrastive learning interferes with the learning of edge / shape filters essential for boundary detection. To investigate this hypothesis further, we jointly train boundary detection with a labeled high-level semantic task. Specifically, we jointly train boundary detection with the ground-truth foreground-background segmentation maps for BSDS500 [Arb+10] from [EH10]. As seen in Table 3.8, the incorporation of semantic information once again does not enhance the single-task performance of boundaries, and even slightly degrades at lower percentage splits.

While CompL yielded performance improvements for monocular depth estimation and semantic segmentation as target tasks, boundary estimation does not observe the same benefits. This further demonstrates the complexity of identifying a universal auxiliary task for all target tasks. Instead, it demonstrates the importance of co-designed self-supervised tasks alongside the downstream task.

Robustness to zero-shot dataset transfer We evaluate the zero-shot dataset transfer capabilities of the BSDS500 [Arb+10] models from Table 3.8 on NYUD-v2 [Sil+12]. Interestingly, even though CompL did not significantly improve the performance in Table 3.8, we find that the robustness experiments in Fig. 3.9 paint a different picture. While MoCo often outperformed DenseCL in Table 3.8, and most methods perform comparatively to the baseline, the additional local constrast of DenseCL significantly improves the robustness experiments. This can be seen from DenseCL consistently outperforming the baseline, as well as all other methods.

Table 3.9: Performance of a multi-task model for monocular depth estimation in RMSE and semantic segmentation in mIoU on NYUD-v2. '+' denote joint training. The multi-task model combined with CompL yields consistent improvements in both tasks.

Model	Depth Labeled Data ↓					Semseg Labeled Data ↑				
	5%	10%	20%	50%	100%	5%	10%	20%	50%	100%
Depth + Semseg	0.997	0.904	0.794	0.665	0.606	10.46	14.99	19.41	26.24	31.66
Depth + Semseg + DenseCL	**0.902**	**0.806**	**0.744**	**0.641**	**0.590**	**10.72**	**15.29**	**20.08**	**28.18**	**33.48**

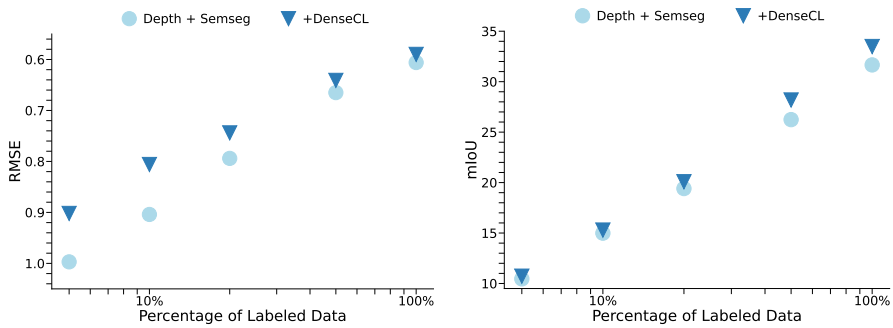
Transfer learning Table 3.8 also reports the performance of the boundary detection transfer learning experiments. All three transfer learning approaches fare worse than ImageNet initialization, corroborating our hypothesis that boundary detection requires representations which are fairly unrelated to the features learned through self-supervision.

3.4.5 *Multi-Task Model (Semseg and Depth)*

Both semantic segmentation (Semseg) and monocular depth estimation (Depth) observed improvements when trained under CompL. In this section, we further investigate the applicability of CompL on MTL models optimized jointly for Depth and Semseg (Depth + Semseg).

Experimental protocol We explore joint training on NYUD-v2 [Sil+12], which provides ground-truth labels for both tasks. We maintain the exact same hyperparameters as the models in Sec. 3.4.2, however, we expect an explicit search could yield additional improvements. No additional task-specific scaling of the losses is used, following [MRK19]. For self-supervised tasks, we only evaluate DenseCL [Wan+21d], as it performed the best for both tasks independently.

Joint optimization Table 3.9 and Fig. 3.10 present the performance of the baseline multi-task model (Depth + Semseg) and the model trained jointly with DenseCL (Depth + Semseg + DenseCL). As in the single-task settings, training under CompL enhances the performance of both Semseg and Depth. Specifically, we again observe a performance gain in every labeled percentage. This demonstrates that, even in the traditional multi-task setting, the additional use of CompL has the potential of yielding further performance gains. In the current setting, Depth observes a noticeable gain over Semseg in low data regimes. This can be attributed to the DenseCL hyperparameters being optimized directly for the improvement



(a) Depth (b) Semseg

Figure 3.10: Performance of (a) monocular depth estimation (Depth) and (b) semantic segmentation (Semseg) on NYUD-v2 for their multi-task model. The multi-task model combined with CompL yields consistent improvements in both tasks.

of Depth. More advanced loss balancing schemes [Che+18b] could yield a redistribution of the performance gains, however, such investigation is beyond the scope of our work.

3.5 CONCLUSION

In this chapter, we introduced CompL, a method that exploits the inductive bias provided by a self-supervised task to enhance the performance of a target task. CompL exploits the label-free supervision of self-supervised methods, facilitating faster iterations through different task combinations. We show consistent performance improvements in fully and partially labeled datasets for both semantic segmentation and monocular depth estimation. While our method eliminated the need for labeling the auxiliary task, it commonly outperforms the traditional MTL with labeled auxiliary tasks on monocular depth estimation. Additionally, the semantic segmentation models trained under CompL yield better robustness on zero-shot cross dataset transfer. We envision our contribution to spark interest in the explicit design of self-supervised tasks for their use in joint training, opening up a new axis of investigation for future multi-task learning research.

4

EFFICIENT VISUAL TRACKING WITH EXEMPLAR TRANSFORMERS

The design of more complex and powerful neural network models has significantly advanced the state-of-the-art in visual object tracking. These advances can be attributed to deeper networks, or the introduction of new building blocks, such as transformers. However, in the pursuit of increased tracking performance, runtime is often hindered. Furthermore, efficient tracking architectures have received surprisingly little attention. In this chapter, we introduce the Exemplar Transformer, a transformer module utilizing a single instance level attention layer for realtime visual object tracking. E.T.Track, our visual tracker that incorporates Exemplar Transformer modules, runs at 47 *FPS* on a CPU. This is up to 8× faster than other transformer-based models. When compared to lightweight trackers that can operate in realtime on standard CPUs, E.T.Track consistently outperforms all other methods on the LaSOT [Fan+19], OTB-100 [WLY13], NFS [Kia+17], TrackingNet [Mul+18], and VOT-ST2020 [Kri+20] datasets.

4.1 INTRODUCTION

Estimating the trajectory of an object in a video sequence, referred to as visual tracking, is one of the fundamental problems in computer vision. Deep neural networks have significantly advanced the performance of visual tracking methods with deeper networks [Ber+16], more accurate bounding boxes [Li+18], or with the introduction of new modules, such as transformers [Yan+21a; Che+21; Wan+21a]. However, these advances often come at the cost of more expensive models. While the demand for realtime visual tracking on applications such as autonomous driving, robotics, and human-computer-interfaces is increasing, efficient deep tracking architectures have received surprisingly little attention. This calls for visual trackers that, while accurate and robust, are capable of operating in realtime under the hard computational constraints of limited hardware.

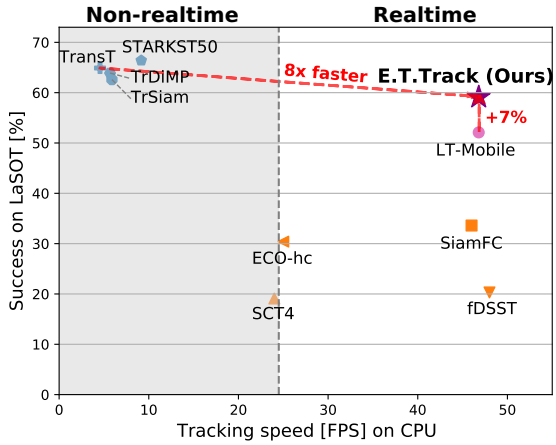


Figure 4.1: Comparison of tracker performance in terms of AUC score (Success in %) on LaSOT vs. tracking speed in *FPS* on a standard CPU. Our Exemplar Transformer Tracker (E.T.Track) outperforms all other realtime trackers. It achieves a 7% higher AUC score than LT-Mobile [Yan+21b]. Furthermore, our approach achieves up to 8× faster runtime on a CPU compared to previous Transformer-based trackers.

Transformers [Vas+17], proposed for machine translation, have also demonstrated superior performance in a number of vision based tasks, including image [Bel+19] and video [Wan+18b] classification, object detection [Car+20], and even multi-task learning [Bru+21]. The field of visual tracking has also observed similar performance benefits [Yan+21a; Sun+20; Wan+21a]. While transformers have enabled the trackers to improve accuracy and robustness, they severely suffer from high computational cost, leading to decreased runtime operation, as depicted in Fig. 4.1. In this work, we set out to find a transformer module, capable increasing tracking accuracy and robustness while not compromising runtime.

In this work we propose Exemplar Attention, a single instance-level attention layer for visual tracking. Our attention module exploits domain specific knowledge to improve the tracker's performance, while maintaining a comparable runtime. Specifically, we build upon two hypotheses. Firstly, one global query value is sufficiently descriptive when tracking a single object. Secondly, a small set of exemplar values can act as a shared memory between the samples of the dataset. Thus, our constrained instance-level Exemplar Attention captures more explicit information about the target object, compared to conventional attention modules.

We develop the Exemplar Transformer Tracker (E.T.Track) by integrating our Exemplar Transformer layer into a Siamese tracking architecture. Specifically, we replace the convolutional layers in the tracker heads with the Exemplar Transformer layer. The additional expressivity from the Exemplar Transformer layer significantly improves the performance of the models based on regular convolutional layers. The added performance gain comes at an insignificant cost in runtime, as seen in Fig. 4.1 when comparing it to the LightTrack-Mobile (LT-Mobile) [Yan+21b]. We further compare our transformer layer for single object tracking to other generic transformer layers. We find that Exemplar Transformer consistently outperforms competing methods, attesting to the benefits of explicitly designing attention layers for the task of visual tracking.

We validate our approach on six benchmark datasets: LaSOT [Fan+19], OTB-100 [WLY13], UAV-123 [MSG16], NFS [Kia+17], TrackingNet [Mul+18] and VOT-ST2020 [Kri+20]. Our proposed tracker runs at 46.8 Frames Per Second (*FPS*) on a CPU, while setting a new state-of-the-art among realtime CPU trackers by achieving 59.1% AUC on the challenging LaSOT dataset.

In summary, our contributions are:

— We introduce the Exemplar Transformer layer, a transformer layer based on a single instance-level attention layer referred to as Exemplar Attention.

— We develop a transformer-based tracking architecture that uses our Exemplar Transformer layer.

— Our tracker runs in realtime on a CPU, while outperforming previous realtime trackers on 5 benchmarks.

4.2 RELATED WORK

Siamese Trackers In recent years, Siamese trackers have gained significant popularity due to their performance capabilities and simplicity. The Siamese-based tracking framework formulates visual object tracking as a template matching problem, utilizing cross-correlation between a search and an image patch. The original work of Bertinetto et al. [Ber+16] introduced SiamFC, the first model incorporating feature correlation into a Siamese framework. Li et al. [Li+18] introduced region proposal networks to increase efficiency and obtain more accurate bounding boxes. More recent advances on the Siamese tracker front include the use of additional

branches [Wan+19], refinement modules for more precise bounding box regression [Yan+21c], and various model update mechanisms [GZX19; Guo+17; YC18; Zha+19c]. Unlike previous Siamese trackers, we propose the Examplar Transformer module that is incorporated into the prediction heads, and improves the tracker's performance at an insignificant runtime increase.

Transformers in Tracking The Transformer [Vas+17] was introduced as a module to improve the learning of long-range dependencies in neural machine translation, by enabling every element to attend to all others. In computer vision, transformers have been used in image [Bel+19] and video [Wan+18b] classification, object detection [Car+20], and even multi-task learning of dense prediction tasks [Bru+21]. More related to our work, transformers have also been utilized to advance the performance of visual trackers. STARK [Yan+21a] utilizes transformers to model the global spatio-temporal feature dependencies between target object and search regions. This is achieved by integrating a dynamically updated template into the encoder, in addition to the regular search and template patch. [Wan+21a] introduced a transformer architecture that improves the standard Siamese-like pipeline by additionally exploiting temporal context. The encoder model mutually reinforces multiple template features by leveraging self-attention blocks. In the decoder, the template and search branch are bridged by cross-attention blocks in order to propagate temporal contexts. [Che+21] also improve Siamese-based trackers by replacing the regular correlation operation by a Transformer-based feature fusion network. The Transformer-based fusion model aggregates global information, providing a superior alternative to the standard linear correlation operation. ToMP [May+22], on the other hand, utilizes a transformer to predicts the weights of a convolutional kernel in order to localize the target in the search region and template patches. In this work, we also design transformer architecture for tracking. Unlike the previous transformers for tracking, Exemplar Transformer is lightweight and can be utilized in computationally limited hardware running at realtime.

Efficient Tracking Architectures With an increase in demand for realtime visual tracking in applications such as autonomous driving, and human-computer-interfaces, efficient deep tracking architectures are essential. Surprisingly, however, little attention has been provided on efficient trackers that can operate on computationally limited hardware. KCF [Hen+14] and fDSST [Dan+16] employ hand-crafted features to enable realtime operation on CPUs. While fast, their reliance on hand crafted features significantly hinders their performance compared to

newer and more complex methods. In contrast, we present an efficient deep tracker that operates at a comparable runtime but performs on par with the more expensive deep trackers. More related to our work, LightTrack [Yan+21b] employs Neural Architecture Search (NAS) to find a lightweight and efficient Siamese tracking architecture. We instead propose an efficient transformer layer that can complement existing architecture advances such as LightTrack. Specifically, our transformer layer can act as a drop in replacement for convolutional layers, increasing performance with negligible effects on runtime.

Efficient Transformers The immense interest in transformer architectures [Dos+21; Par+18; Car+20; Wan+18b] resulted in the development of various efficient model variants that can be grouped in 4 major categories [Tay+20]. *Low Rank/ Kernel* methods assume and leverage low-rank approximations of the self-attention matrix [Kat+20; Cho+20]. *Memory/ Downsampling* methods learn a side memory module to access multiple tokens simultaneously, or simply reduce the sequence length [Tan+22; ZY21; Liu+21]. *Fixed/ Factorized/ Random Patterns* limit the field of view of the self-attention, such as using block patterns [Ryo+21; ZY21; Liu+21; Ram+19]. *Learnable Patterns* replace the fixed pattern, as in standard transformers, with dynamic patterns [VKF20; Wan+20b; KKL20]. Our work falls in the intersection of *Memory/ Downsampling* and *Fixed/ Factorized/ Random Patterns*. Unlike the aforementioned works that aim to design generic attention layers, Exemplar Attention is instead designed for the task of single object visual tracking by exploiting domain specific knowledge.

4.3 EFFICIENT TRACKING WITH TRANSFORMERS

Striking a balance between well performing object trackers and runtime speeds that fall in the realtime envelope, is a challenging problem when deploying on computationally limited devices. In this section, we introduce the Exemplar Transformer, a transformer architecture based on single instance level attention layers for single object tracking. While lightweight, our Exemplar Transformer significantly closes the performance gap with the computationally expensive transformer-based trackers [Yan+21a; Che+21; Wan+21a]. Sec. 4.3.1 first presents the original Transformer of Vaswani *et al.* [Vas+17], followed by our Exemplar Transformer formulation. Sec. 4.3.2 introduces our E.T.Track. Specifically, it first outlines the overall architecture, and presents how Exemplar Transformers are utilized within the tracker.

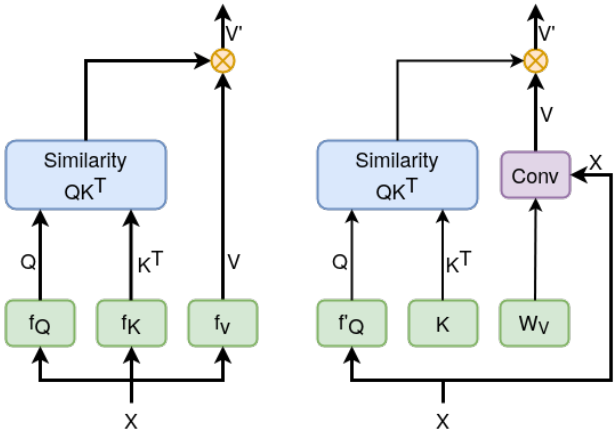


Figure 4.2: Comparison of the standard scaled dot-product attention module [Vas+17] (left) with our Exemplar Attention module (right). The matching blocks are indicated by identical colours. The line thickness is indicative of the tensor size.

4.3.1 *Exemplar Transformers*

Standard Transformer The Transformer [Vas+17], introduced for machine translation, receives a one dimensional input sequence $x \in \mathbb{R}^{N \times D}$ with N feature vectors of dimensions D. The input sequence is processed by a series of transformer layers defined as

$$T(x) = f(A(x) + x). \tag{4.1}$$

The function $f(\cdot)$ is a lightweight Feed-Forward Network (FFN) that projects independently each feature vector. The function $A(\cdot)$ represents a self-attention layer that acts across the entire sequence. Specifically, the authors used the "Scaled Dot-Product Attention" depicted on the left of Fig. 4.2, and defined as

$$\begin{aligned} A(x) &= \text{softmax}\left(\frac{\overbrace{Q}^{f_Q(x)}\overbrace{K^T}^{f_K(x)}}{\underbrace{\sqrt{d_k}}_{\text{constant}}}\right)\overbrace{V}^{f_V(x)} \\ &= \text{softmax}\left(\frac{\overbrace{(xW_Q)}^{f_Q(x)}\overbrace{(W_K^T x^T)}^{f_K(x)}}{\underbrace{\sqrt{d_k}}_{\text{constant}}}\right)\overbrace{(xW_V)}^{f_V(x)}. \end{aligned} \tag{4.2}$$

The queries $Q \in \mathbb{R}^{N \times D_{QK}}$, keys $K \in \mathbb{R}^{N \times D_{QK}}$, and values $V \in \mathbb{R}^{N \times D_V}$ represent projections of the input sequence, while $\sqrt{d_k}$ is a normalization constant. The self attention, therefore, computes a similarity score between all representations, linearly combines the feature representations, and accordingly adapts the input representation x in Eq. 4.1. The computational complexity of Eq. 4.2 is $\mathcal{O}(N^2 D)$, *i.e.* it scales quadratically with the length of the input sequence.

Exemplar Attention We now introduce Exemplar Attention, the key building block of the Exemplar Transformer module. We hypothesize that, while the direct connection between all features is essential in machine translation and some vision tasks, this design choice can be sub-optimal when attending to the single object being tracked. We describe the required modifications of the individual components below.

The standard Query function f_Q projects every spatial location of the feature map independently to a query space. Unlike machine translation where every feature represents a specific word or token, adjacent spatial representation in vision tasks often correspond to the same object. Consequently, we aggregate the information of the feature map $X \in \mathbb{R}^{H \times W \times D}$, where $H \times W$ represents the spatial dimensions. Specifically, we use a 2D adaptive average pooling layer with an output spatial dimension $S \times S$, followed by a flattening operation. The operation is denoted $\Psi_S(X)$, decreasing the output spatial dimension to S^2. The compressed representation of X is then projected to a query space as in the standard self-attention formulation.

$$Q = \Psi_S(X) W_Q \in \mathbb{R}^{S^2 \times D_{QK}} \tag{4.3}$$

We hypothesize that for single instance tracking, one global query value is sufficient to identify the object of interest, while also decreasing the computational complexity of the module. To this extent, we set $S = 1$. This design choice is further supported by the success of global pooling in classification architectures [He+16], as well as transformer based object detection [Car+20].

The keys and values, as presented in Eq. 4.2, are per spatial location linear projections of the input. The self-attention layer then enables the learning of spatial correlations, at the cost of every feature attending to all others. This eliminates spatial biases built into convolutional layers. Rather than requiring a fine grained feature map and relying solely on intra-sample relationships, we instead learn a small set of exemplar representations. The exemplar representations encapsulates dataset information in order to dynamically adapt the attention layer

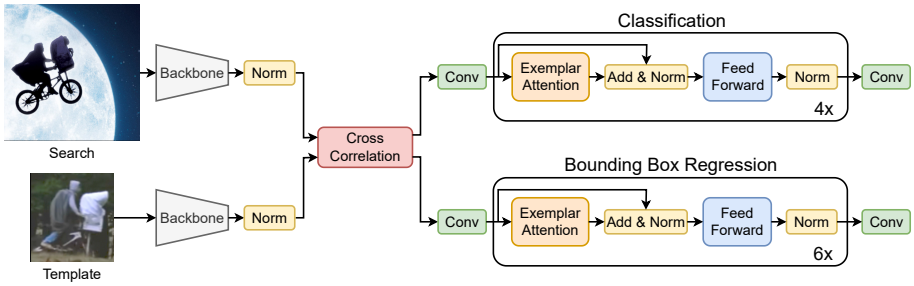


Figure 4.3: E.T.Track - a Siamese tracking pipeline that incorporates Exemplar Transformers in the tracker head.

given the global query token and the captured information. To this end, we optimize a small set of exemplar keys $K = W_K \in \mathbb{R}^{E \times D_{QK}}$ that, unlike the formulation in Eq. 4.2, are independent of the input. The similarity matrix therefore associates the global query, Eq. 4.3, to exemplars. Our attention layer then refines the input representation on the local level by replacing the projection $f_V(\cdot)$ with a convolutional operation

$$V = W_V \circledast X \in \mathbb{R}^{E \times H \times W \times D_V}, \tag{4.4}$$

where $W_V \in \mathbb{R}^{E \times Z \times Z}$ can be of any spatial dimension Z, while the number of exemplars E can be chosen arbitrarily. We use $E = 4$ in our experiments, which is significantly smaller than the dimensions $H \times W$, maintaining comparable runtime.

Our efficient Exemplar Attention is therefore defined as,

$$A(x) = \text{softmax}\left(\frac{\overbrace{(\Psi_S(X)W_Q)}^{f_Q(x)}\overbrace{(W_K^T)}^{f_K(\cdot)}}{\underbrace{\sqrt{d_k}}_{\text{constant}}}\right)\overbrace{(W_V \circledast X)}^{f_V(x)}, \tag{4.5}$$

but can also be written as,

$$A(x) = \left[\text{softmax}\left(\frac{(\Psi_S(X)W_Q)(W_K^T)}{\sqrt{d_k}}\right)W_V\right] \circledast X. \tag{4.6}$$

Exemplar Attention, while inspired by the scaled dot-product attention, is conceptually very different. In self-attention (4.2), $f_{\{Q,K,V\}}$ act as projections to their corresponding feature spaces, with the similarity function learning the relationships between all spatial locations. In other words, self-attention relies solely on intra-sample relationships, and

Algorithm 4.1 Pseudocode of the Exemplar Attention layer, Eq. 4.6.

function `ExemplarAttention(`X`):`
- $Q \leftarrow \Psi_S(X)W_Q$ Eq. 4.3
- $K \leftarrow W_K$
- $V \leftarrow W_V$
- $\text{sim} \leftarrow \text{softmax}(Q \cdot K^T)$
- $\text{sim} \leftarrow \text{sim}/\sqrt{d_k}$
- $W_A = \text{sim} \cdot V$
- $A(X) \leftarrow W_A \circledast X$
- **return** $A(X)$

end

therefore requires fine-grained representations. Instead, the Exemplar Attention layer enforces the attending over a single instance through the use of a global query token. The global query encapsulated the representation of the object, is dynamically generated from the input image, and applied locally on the feature map using a convolutional operation. To enable the use of a single query token, we exploit dataset information to form the exemplar representations through end-to-end optimization, eliminating the need of the intra-sample similarity function. A visual comparison between the two attention mechanisms is depicted in Fig. 4.2, while the pseudocode for the Exemplar Attention layer (Eq. 4.6) can be seen in Algorithm 4.1.

4.3.2 *E.T.Track Architecture*

In this section we introduce the base tracking architecture used throughout our work. While Exemplar Transformers can be incorporated into any tracking architecture, we evaluate its efficacy on lightweight Siamese trackers. An overview of the E.T.Track architecture can be seen in Fig. 4.3.

Our model employs the lightweight backbone model of Yan et al. [Yan+21b], LT-Mobile. The model was identified by NAS on a search space consisting of efficient and lightweight building blocks. The feature extracting backbone consists of 3×3 convolutional layers, depthwise separable convolutional layers and mobile inverted bottleneck layers with squeeze and excitation modules.

The Exemplar Transformer layer can act as a drop in replacement for any convolution operation of the architecture. We replace all the convolutions in the classification and bounding box regression branches, while keeping the lightweight backbone architecture untouched. This eliminates the need for retraining the backbone on ImageNet [Den+09].

Search and template frames are initially processed through a backbone network. The similarity between the representations is computed by a pointwise cross-correlation. The resulting correlation map is then fed into the tracker head, where it is processed by a classification branch and a bounding box regression branch in parallel. The bounding box regression branch predicts the distance to all four sides of the bounding box. The classification branch predicts whether each region is part of the foreground or the background. During training, the bounding box regression branch considers all the pixels within the ground truth bounding box as training samples, therefore, the model is able to determine the exact location of the object even when only small parts of the input image are classified as foreground. The model is trained by optimizing a weighted combination of the binary cross-entropy (BCE) loss and the IoU loss [Yu+16] between the predicted and ground-truth bounding boxes. For more details, as well as more information on the data preprocessing, we refer the reader to [Zha+20].

4.4 EXPERIMENTS

We first present implementation details of our tracker in Sec. 4.4.1. The comparison to state-of-the-art is presented in Sec. 4.4.2, followed by an ablation study in Sec. 4.4.5. Code and trained models will be released on publication.

4.4.1 *Implementation Details*

Architecture We adopt the LT-Mobile architecture of Yan et al. [Yan+21b] as our baseline due to its performance versus efficiency trade-off. LT-Mobile is comprised of a small encoder, and later branches to the classification and regression heads. The classification head is comprised of 6 convolutional modules, while the regression head is comprised of 8. Each convolutional module consists of a Depthwise Separable Convolution [How+17], a Batch Normalization layer [IS15] and a Rectified Linear unit. E.T.Track replaces each decoder convolutional module with an Exemplar Transformer Layer, introduced in Sec. 4.3.1. The learn-

able "value" parameters of the attention module are initialized using kaiming initialization [He+15], while the learnable "keys" parameters are initialized using a normal distribution. The FFN consists of 2 linear layers with a ReLU activation, dropout [Sri+14] with a ratio of 0.1, and LayerNorm [BKH16].

Training All models have been trained using an Nvidia GTX TITAN X, and evaluated on an Intel(R) Core(TM) i7-8700 CPU @ 3.20GHz. The training of our E.T.Track architecture is based on the training framework used in LightTrack [Yan+21b] which, in turn, is based on OCEAN [Zha+20]. As is common practice, we initialize the backbone with ImageNet pre-trained weights. The models are optimized using stochastic gradient decent [Rud16] with a momentum of 0.9, and a weight decay of 1e-4 for 50 epochs. During the first 10 epochs, the backbone parameters remain frozen. We use a step learning rate scheduler during a warmup period of 5 epochs, increasing the learning rate from 2e-2 to 1e-1, followed by a logarithmically decreasing learning rate from 1e-1 to 2e-4 for the remainder. We utilize 3 GPUs and sample 32 image pairs per GPU for each batch. The sampled image pairs consist of a 256×256 search frame and a 128×128 template frame, sampled from training splits of LaSOT [Fan+19], TrackingNet [Mul+18], GOT10k [HZH19] and COCO [Lin+14]. Specifically, the two frames are sampled within a range of 100 frames for LaSOT [Fan+19] and GOT10k [HZH19], 30 frames for TrackingNet [Mul+18], and 1 frame for COCO [Lin+14]. Both patches are further shifted and scaled randomly.

4.4.2 *Comparison to State-of-the-Art*

We compare our proposed E.T.Track to state-of-the-art methods on 7 benchmarks: OTB-100 [WLY13], NFS [Kia+17], UAV-123 [MSG16], LaSOT [Fan+19], TrackingNet [Mul+18], VOT-ST2020 [Kri+20], and VOT-RT2020 [Kri+20]. Specifically, we evaluate transformer-based trackers [Che+21; Wan+21a; Yan+21a], realtime CPU trackers [Dan+17; Yan+21b], as well as additional seminal trackers [Dan+19; Li+19a; Bha+19; DGT20; Voi+20]. We benchmark runtimes on CPU due to their wide incorporation on computationally limited platforms, and report them in *FPS*. However, both E.T.Track and the other methods do not rely on CPU specific modules and can therefore be used on light weight GPUs for further latency improvements, such as the NVIDIA Jetson.

LaSOT [Fan+19] The LaSOT dataset is highly challenging, and includes very long sequences with an average of 2500 frames per sequence.

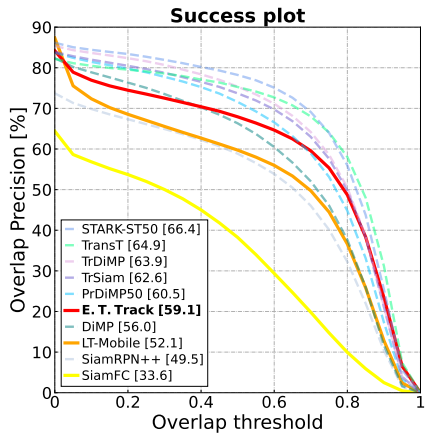


Figure 4.4: Success plot on the LaSOT dataset. The CPU realtime trackers are indicated by continuous lines in warmer colours, while the non-realtime trackers are indicated by dashed lines in colder colours. E.T.Track significantly outperforms the other realtime trackers, and even outperforms some of the more established trackers such as DiMP [Bha+19]. Furthermore, it significantly closes the performance gap with the more expensive transformer trackers.

Therefore, robustness is essential to achieving a high score. The success plot in Fig. 4.4 depicts the CPU realtime trackers with warmer colour continuous lines, while the non-realtime trackers are indicated by dashed lines in colder colours. Unlike online-learning methods such STARK that utilize a dynamically updated template, our model only uses the features of the template patch extracted in the first sequence of the frames. Even so, our model is very robust and reaches an AUC score of 59.1%, outperforming the popular DiMP tracker [Bha+19] by 2.2%. Compared to the lightweight mobile architecture of LT-Mobile [Yan+21b], our model improves the success score by an astonishing 7% while achieving a comparable speed.

NFS [Kia+17] We additionally evaluate our approach on the NFS dataset that contains fast-moving objects. The results are presented in Table 4.1 and Fig. 4.5a. E.T.Track reaches an AUC score of 59%, outperforms all realtime trackers by at least 3.7%.

OTB-100 [WLY13] OTB-100 contains 100 sequences. As shown in Table 4.1 and Fig. 4.5b, the current state-of-the-art is achieved by the recently introduced TrDiMP [Wan+21a] with an AUC score of 71.1%.

Our model achieves an AUC score of 67.8%, marking it as the best performing realtime tracker.

UAV-123 [MSG16] UAV-123 contains a total of 123 sequences from aerial viewpoints. The AUC results are shown in Table 4.1 and Fig. 4.5c. Unlike the other datasets, E.T.Track performs comparably to LT-Mobile, with a performance of 62.3%.

TrackingNet [Mul+18] We further evaluate the trackers on the 511 sequences of the TrackingNet test set, and report the results in Table 4.2. Similarly to the other datasets, E.T.Track outperforms all other realtime trackers. Specifically, E.T.Track improves LT-Mobile precision by 1.05%, normalized precision by 2.42%, and AUC by 2.48%. Comparing E.T.Track to more complex transformer-based trackers such as TrSiam [Wan+21a], our model is only 2.2% worse in terms of precision, 2.32% in terms of normalized precision, and 3.12% in terms of AUC while running almost 8× faster on a CPU. This further demonstrates that, while transformers have the capability to significantly improve performance, transformer modules do not need to be prohibitively expensive for computationally constrained devices to achieve most of the performance gains.

VOT-ST2020 [Kri+20] We also evaluate bounding box predicting trackers on the anchor-based short-term tracking dataset of VOT-ST2020. Unlike other tracking datasets, VOT2020 contains various anchors that are placed Δ_{anc} frames apart. The trackers are evaluated in terms of Accuracy, Robustness and Expected Average Overlap (EAO). Accuracy represents a weighted combination of the average overlap between the ground truth and the predicted target predictions on subsequences defined by anchors. Robustness indicates the percentage of frames before the trackers fails on average. Finally, EAO is a measure for the overall tracking performance and combines accuracy and robustness. The results are shown in Table 4.3. While our model outperforms the lightweight convolutional baseline model introduced in [Yan+21b] by $1-2\%$ in terms of accuracy and robustness, the largest performance increase can be noted in terms of robustness, where the performance is increased by 5.2%. We find that learning exemplar representations from the dataset coupled with an image-level query representation significantly increase the tracker's robustness compared to its convolutional counterpart.

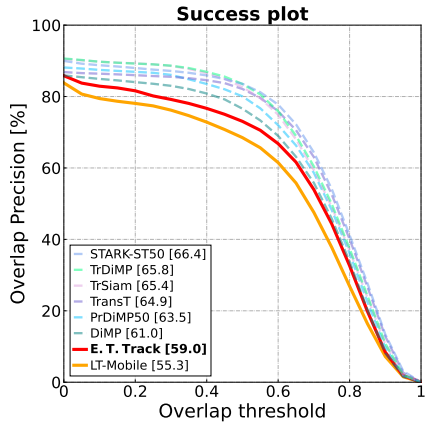
VOT-RT2020 [Kri+20] Finally, similar to VOT-ST2020, we also evaluate VOT-RT2020 of Kristan et al. [Kri+20]. The results are presented in Table 4.4. While the performance of our model is comparable to LT-Mobile [Yan+21b] in terms of accuracy, our model is nearly 6%

Table 4.1: State-of-the-art comparison on the NFS, OTB-100 and UAV-123 datasets in terms of Area Under the Curve (AUC). The best score is highlighted in blue while the best realtime score is highlighted in red. We additionally report CPU runtime speeds in *FPS*.

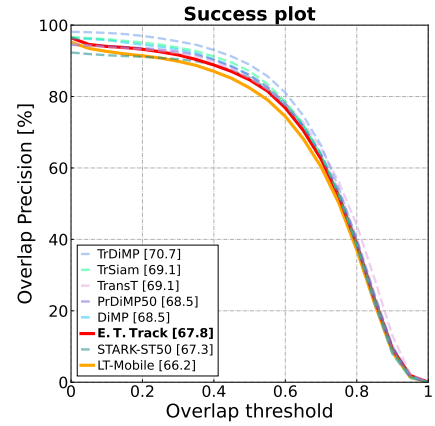
	non-realtime									realtime		
	ATOM [Dan+19]	SiamRPN++ [Li+19a]	DiMP-50 [Bha+19]	PrDiMP-50 [DGT20]	SiamR-CNN [Voi+20]	TransT [Che+21]	TrDiMP [Wan+21a]	TrSiam [Wan+21a]	STARK-ST50 [Yan+21a]	ECO [Dan+17]	LT-Mobile [Yan+21b]	**E.T.Track (Ours)**
NFS	58.4	50.2	62	63.5	63.9	65.7	66.5	65.8	66.4	46.6	55.3	**59.0**
UAV-123	64.2	61.3	65.3	68	64.9	69.4	67.5	67.4	68.8	51.3	**62.5**	62.3
OTB-100	66.9	69.6	68.4	69.6	70.1	69.1	71.1	70.8	67.3	64.3	66.2	**67.8**
CPU Speed	20	15	15	15	15	5	6	6	9	25	47	47

Table 4.2: State-of-the-art comparison on the TrackingNet test set, comprised of 511 sequences. The trackers are compared in terms of Precision (Prec.), Normalized Precision (N. Prec.), and Success. The best score is highlighted in blue while the best realtime score is highlighted in red. We additionally report CPU runtime speeds in *FPS*.

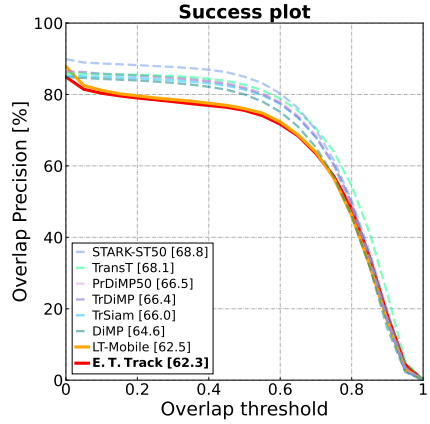
	non-realtime									realtime		
	ATOM [Dan+19]	SiamRPN++ [Li+19a]	DiMP-50 [Bha+19]	PrDiMP-50 [DGT20]	SiamR-CNN [Voi+20]	TransT [Che+21]	TrDiMP [Wan+21a]	TrSiam [Wan+21a]	STARK-ST50 [Yan+21a]	ECO [Dan+17]	LT-Mobile [Yan+21b]	**E.T.Track (Ours)**
Prec. (%)	64.84	69.38	68.7	70.4	80	80.3	73.1	72.7	-	48.86	69.5	**70.6**
N. Prec. (%)	77.11	79.98	80.1	81.6	85.4	86.7	83.3	82.9	86.1	62.14	77.9	**80.3**
Success (%)	70.34	73.3	74	75.8	81.2	81.4	78.4	78.1	81.3	56.13	72.5	**75.0**
CPU Speed	20	15	15	15	15	5	6	6	9	25	47	47



(a) Success plot on the NFS dataset. Our tracker outperforms LT-Mobile by a significant margin.



(b) Success plot on the OTB-100 dataset. Our tracker outperforms LT-Mobile by a small margin.



(c) Success plot on the UAV-123 dataset. The performance of our tracker is comparable to the performance of LT-Mobile.

Figure 4.5: Success plots. The CPU realtime trackers are indicated by continuous lines in warmer colours, while the non-realtime trackers are indicated by dashed lines in colder colours. For efficient trackers, we limited our comparison to LT-Mobile [Yan+21b], as SiamRPN++ [Li+19a] and SiamFC [Ber+16] were consistently outperformed by a large margin.

better in terms of robustness. Similarly to VOT-ST2020, we find that learning exemplar representations from the dataset coupled with an image-

Table 4.3: Comparison of bounding box predicting trackers on the VOT-ST2020 dataset. We report the Expected Average Overlap (EAO), the Accuracy and Robustness. The best score is highlighted in blue while the best realtime score is highlighted in red. We additionally report CPU runtime speeds in *FPS*.

	non-realtime					realtime		
	SiamFC [Ber+16]	ATOM [Dan+19]	DiMP [Bha+19]	SuperDiMP [Dan+]	STARK-ST50 [Yan+21a]	KCF [Hen+14]	LT-Mobile [Yan+21b]	E.T.Track **(Ours)**
EAO	0.179	0.271	0.274	0.305	0.308	0.154	0.242	**0.267**
Accuracy	0.418	0.462	0.457	0.477	0.478	0.407	0.422	**0.432**
Robustness	0.502	0.734	0.740	0.786	0.799	0.432	0.689	**0.741**
CPU Speed	6	20	15	15	9	95	47	47

Table 4.4: Comparison of bounding box predicting trackers on the VOT-RT2020 dataset. We report the Expected Average Overlap (EAO), the Accuracy and Robustness. The best score is highlighted in blue while the best realtime score is highlighted in red. We additionally report CPU runtime speeds in *FPS*.

	non-realtime				realtime		
	SiamFC [Ber+16]	ATOM [Dan+19]	DiMP [Bha+19]	SuperDiMP [Dan+]	KCF [Hen+14]	LT-Mobile [Yan+21b]	**E.T.Track (Ours)**
EAO	0.172	0.237	0.241	0.289	0.154	0.217	**0.227**
Accuracy	0.422	0.440	0.434	0.472	0.406	0.418	**0.418**
Robustness	0.479	0.687	0.700	0.767	0.434	0.607	**0.663**
CPU Speed	6	20	15	15	95	47	47

level query representation significantly increase the tracker's robustness compared to its convolutional counterpart.

4.4.3 *Attributes Analysis*

Table 4.5 presents the results of various trackers on different sequence attributes of the LaSOT dataset [Fan+19]. We consistently outperform the other realtime trackers by a significant margin in every attribute. The attribute with the largest performance gains compared to LT-Mobile [Yan+21b] are *Full Occlusion* with 10.4%, *Motion Blur* with 9.7%, *Background Clutter* with 8.5%, and *Fast Motion* with 7.5%. These attributes are either known limitation of the tracking pipeline utilized [Zha+20], as discussed in Sec. 4.4.4, or can benefit from increased network capacity. We find that the incorporation of our Exemplar Transformer layers increases robustness and improves attributes that are even known limitations of the overall framework.

Table 4.5: LaSOT attribute-based analysis. Each column corresponds to the results computed on all sequences in the dataset with the corresponding attribute. The trackers that do not run in real-time are highlighted in grey. The overall best score is highlighted in blue while the best realtime score is highlighted in red.

	Illumination Variation	Partial Occlusion	Deformation	Motion Blur	Camera Motion	Rotation	Background Clutter	Viewpoint Change	Scale Variation	Full Occlusion	Fast Motion	Out-of-View	Low Resolution	Aspect Ration Change	Total
STARK-ST50	66.8	64.3	66.9	62.9	69.0	66.1	57.3	67.8	66.1	58.7	53.8	62.1	59.4	64.9	66.4
TransT	65.2	62.0	67.0	63.0	67.2	64.3	57.9	61.7	64.6	55.3	51.0	58.2	56.4	63.2	64.9
TrDiMP	67.5	61.1	64.4	62.4	68.1	62.4	58.9	62.8	63.4	56.4	53.0	60.7	58.1	62.3	63.9
TrSiam	63.8	60.1	63.8	61.1	65.5	62.0	55.1	60.8	62.5	54.5	50.6	58.9	56.0	61.2	62.6
PrDiMP50	63.3	57.1	61.3	58.0	64.0	59.1	55.4	61.7	60.1	51.6	49.2	57.0	54.8	59.0	60.5
DiMP	59.5	52.1	56.6	54.6	59.3	54.5	49.7	56.7	55.8	47.5	45.6	49.5	49.1	54.5	56.0
SiamRPN++	53.0	46.6	52.8	44.2	51.3	48.5	44.9	44.4	49.4	36.6	31.6	41.6	38.5	47.2	49.5
LT-Mobile	55.0	48.9	57.3	45.8	52.9	51.2	43.3	49.9	51.9	38.3	33.6	43.7	40.8	49.9	52.1
SiamFC	34.6	30.6	35.1	30.8	33.3	31.0	30.8	28.6	33.2	24.5	19.5	25.6	25.2	30.8	33.6
E.T.Track (Ours)	**61.3**	**55.7**	**61.0**	**55.5**	**60.2**	**58.1**	**51.8**	**55.9**	**58.8**	**48.7**	**41.1**	**51.1**	**48.8**	**56.9**	**59.1**

Table 4.6: Direct per-sequence comparison of E.T.Track and LT-Mobile [Yan+21b] on various sequences in terms of Average Overlap (AO). The best performance is highlighted in blue.

	Dataset	LT-Mobile [Yan+21b]	**E.T.Track (Ours)**
person8-2	UAV-123 [MSG16]	0.889	**0.915**
Human7	OTB [WLY13]	0.813	**0.883**
boat-9	UAV-123 [MSG16]	0.483	**0.803**
basketball-3	NFS [Kia+17]	0.259	**0.707**
drone-2	LaSOT [Fan+19]	0.192	**0.887**

When comparing our model to the non-realtime state-of-the-art STARK [Yan+21a], our model observes an average performance drop of -7.3%. The most challenging attributes are *Viewpoint Change*, *Full Occlusion*, *Fast Motion*, *Out-of-View*, and *Low Resolution*. This analysis paves the path for future research in the design of novel modules for efficient tracking, specifically tackling the identified challenging attributes.

4.4.4 *Video Visualizations*

We additionally provide sequence comparisons between E.T.Track and LT-Mobile [Yan+21b]. Table 4.6 lists the sequences compared, and reports their performance. The videos can be found in the supplementary folder of the associated paper [Bla+23].

person8-2 The person8-2 sequence of the UAV-123 dataset [MSG16] of a man running on grass nicely demonstrates that our tracker does not lose track of the target even when he partially moves out of the frame. Specifically, E.T.Track is able to completely recover when the target moves back into the frame. LT-Mobile [Yan+21b] yields comparable results.

Human7 Human7 from the OTB dataset [WLY13] films a woman walking. Even though the video appears to be jittery, the appearance and shape of the target object changes only marginally. Our model achieves an average overlap of 88% which is 7% higher than LT-Mobile [Yan+21b].

boat-9 The boat-9 from the UAV-123 dataset [MSG16] depicts a target which not only changes appearance, but also significantly decreases in size due to an increasing distance to the camera. We find that E.T.Track can still handle such scenarios, and unlike LT-Mobile, it maintains track of the boat even after a 180-degree turn. E.T.Track is therefore more

robust than LT-Mobile, attributed to the increased capacity introduced by the Exemplar Transformer layers.

basketball-3 In the basketball-3 sequence of NFS [Kia+17], the increased robustness introduced by the Exemplar Transformer layer enables the separation between the player's head and the basketball, unlike LT-Mobile.

drone-2 The drone-2 sequence of LaSOT [Fan+19] shows a target that shortly moves completely out of the frame, and later re-enters the scene with a different appearance to the initial frame. Furthermore, the target object's location deviates from the tracker's search range when re-entering the scene. These two aspects pose a challenge both for our model, as well as LT-Mobile [Yan+21b], and are inherent limitations of the tracking inference pipeline used in both approaches [Zha+20]. Specifically, the tracking pipeline contains a post-processing step in which the predicted bounding boxes are refined. Changes in size, as well as changes in the bounding box aspect ratios, are therefore penalized. In addition, both models search only within a small image patch around the previously predicted target location. This challenge can potentially be addressed by integrating our Exemplar Transformer layer into trackers that directly predict bounding boxes without any post-processing. We did not investigate this further, but consider this an interesting direction for future research.

4.4.5 *Ablation Study*

To further understand the contributions of the different components, we conduct a number of controlled experiments on three datasets. Specifically, we report AUC on OTB-100 [WLY13], NFS [Kia+17], and LaSOT [Fan+19].

Baseline We commence our ablation study from the mobile architecture of [Yan+21b], LT-Mobile, due to its performance versus efficiency trade-off. We refer to LT-Mobile, our baseline Siamese tracker, as the Convolutional (Conv) baseline. In LT-Mobile, the similarity of the search and template patch features is computed by a pointwise cross-correlation. The feature map is then passed to the tracker head consisting of two branches, the classification and the bounding box regression branches, as explained in Sec. 4.4.1. The performance of the baseline model, Conv, is reported in Table 4.7.

Exemplar Attention We first evaluate the efficacy of the Exemplar Attention as a drop in replacement for the convolutional layer. We

Table 4.7: Ablating the different components of the Exemplar Transformer module in terms of AUC on NFS, OTB, and LaSOT datasets. Conv refers to LT-Mobile [Yan+21b] that acts as our convolutional baseline. We evaluate the Exemplar Attention (Att) module, Feed-Forward Network (FFN), and Template Conditioning (T-Cond). The best score is highlighted in blue. The final model, depicted in Fig. 4.3 includes the Att and FFN modules.

Components				Tasks		
Conv	Att	FFN	T-Cond.	NFS	OTB-100	LaSOT
✓				55.3	66.2	52.1
	✓			56.6	65.8	53.6
	✓	✓		58	67.3	59.1
	✓	✓	✓	59.0	66.9	57.9

replace the convolutional layers with the Exemplar Attention, followed by a residual connection and a normalization layer. In other words, setting the FFN ($f(\cdot)$) in Eq. 4.1 to identify. We report the performance of the Attention (Att) module in Table 4.7. The performance on NFS increases by 1.3%, and on LaSOT by 1.5%, demonstrating the effectiveness of our Exemplar Attention module. We note that this performance increase is without the use of the FFN, a key design choice in the transformer architectures [Vas+17].

FFN Similar to the original Transformer architecture [Vas+17], we evaluate the effect of additionally using a lightweight FFN followed by a LayerNorm layer. We find that the additional expressivity introduced by the FFN improves the performance on all three datasets, as seen in Table 4.7. The highest performance increase is achieved on LaSOT, where the AUC score increases by 5.5%. This yields our final E.T.Track model, depicted in Fig. 4.3.

Template conditioning The queries used in the Exemplar Transformer so far are solely based on a transformed version of the initial correlation map. We further explore the impact of incorporating template information into our Exemplar Attention module. Specifically, we average pool the feature map corresponding to the template patch, and sum the representation to each layer's input. As seen from the Template Conditioning (T-Cond) experiments in Table 4.7, the richer queries lead to an improvement on NFS. However, on OTB-100 and LaSOT, the model did not benefit from the additional information. To this extend, we decide to not use the T-Cond module in our final module, keeping our final model simpler.

Table 4.8: Effect of the number of exemplars (-Ex) in terms of AUC on NFS, OTB, and LaSOT datasets. Conv refers to LT-Mobile [Yan+21b] that acts as our convolutional baseline. The best score is highlighted in blue. The final model makes use of four exemplars.

	Conv	1-Ex	4-Ex	16-Ex
NFS	55.3	57.6	58.0	58.0
OTB-100	66.2	66.5	67.3	66.1
LaSOT	52.1	57.2	59.1	57.4

Number of Exemplars Table 4.8 reports the performance given the number of Exemplars. While more Exemplars increases the overall capacity of the model, and as such, one would expect further performance gains, our experiments yield different results. Specifically, 4 Exemplars yield consistently better results across all the datasets. We hypothesize that, while training a model with a larger number of experts can increase performance, modifications in the optimization process are required to ensure the selection of the appropriate exemplar. The efficient implementation of the Exemplar Attention module, Eq. 4.6, ensures a comparable runtime even with a larger number of exemplars.

Interestingly, while single Exemplar Attention is mathematical equivalent to a regular convolution with a residual operation, the additional FFN layer following the Exemplar Attention increases considerably the performance. Specifically, we observe a performance increase of 2.3% on NFS, 0.3% on OTB-100, and 5.1% on LaSOT.

Number of Query Vectors As discussed in Sec. 4.3.1, we set $S = 1$ in our experiments based on the assumption that one global token encapsulates sufficient information for the task of single object tracking. To further evaluate this hypothesis, we ablate the parameter S. Specifically, the input feature map is divided into $S \times S$ patches, for which we compute individual query vectors. Table 4.9 presents the results of our experiments and validates our assumption that utilizing a single token as a global representation yields the best results.

Backbone Alternatives All experiments reported so far utilize the LT-Mobile encoder. To demonstrate the flexibility of Exemplar Transformers, as well as their independence to the encoder architecture, we evaluate the use of different encoder architectures. Specifically, we compare the performance of the two tracker head module variants (Convolution, Exemplar Transformer) in combination with ShuffleNet [Zha+18a], MobileNetV3 [How+19], ResNet-18 [He+16], and LT-Mobile [Yan+21b]. The

Table 4.9: Ablation experiment of the different values for S reported in terms of AUC on NFS, OTB, and LaSOT datasets. The best score is highlighted in blue. The final model makes use of a single global query vector.

	S=1	S=2	S=4
NFS	59.0	46.6	46.7
OTB-100	67.8	55.5	57.5
LaSOT	59.1	43.7	42.6

Table 4.10: Comparison of Exemplar Transformer (E.T.) and Convolutional (Conv) modules on different backbone models in terms of AUC on NFS, OTB, and LaSOT datasets. Conv refers to LT-Mobile [Yan+21b] that acts as our convolutional baseline. The best score is highlighted in blue. The E.T. consistently outperforms the Conv models, independently of the backbone used.

	ShuffleNet [Zha+18a]		MobileNetV3 [How+19]		ResNet-18 [He+16]		LT-Mobile [Yan+21b]	
Conv	✓		✓		✓		✓	
E.T. (Ours)		✓		✓		✓		✓
NFS	54.9	56.2	56.8	56.8	55.8	57.3	55.3	59.0
OTB-100	61.3	61.8	64.5	65.3	65.3	65.7	66.2	67.8
LaSOT	48.6	49.8	52.1	52.7	55.9	56.5	52.1	59.1

results presented in Table 4.10 demonstrate consistent performance gains independent of the encoder architecture, highlighting the superiority of our Exemplar Transformer to its convolutional counterpart.

Comparison of Alternative Transformer Layers To validate the design choices and hypothesis that lead to the Exemplar Transformer module, we additionally compare to other Transformer Layer variants. All Transformer layers evaluated can also act as drop-in replacements to standard convolutions. Specifically, we evaluate the Standard [Vas+17], Clustered [VKF20], Linear [Kat+20], Local [Ram+19], and Swin [Liu+21] Transformers. The selection ensures at least one method from every transformer category defined in Sec. 4.2, while using their official public implementations ensures a fair comparison. The results in Table 4.11 demonstrate that our Exemplar Transformer (E.T.) consistently outperforms all other attention variants across all the datasets. These findings further validate our hypothesis that one global query and a small set

Table 4.11: Comparison of the convolutional baseline (Conv) and the different attention modules in terms of AUC on NFS, OTB, and LaSOT datasets. The best score is highlighted in blue. E.T.Track consistently outperforms all other transformer variants.

	Conv [Yan+21b]	Standard [Vas+17]	Clustered [VKF20]	Linear [Kat+20]	Local [Ram+19]	Swin [Liu+21]	E.T.Track (Ours)
NFS	55.3	55.3	57.5	55.8	55.8	55.4	59.0
OTB-100	66.2	65.3	67.5	65.4	64.8	64.2	67.8
LaSOT	52.1	54.2	56.5	53.5	53.4	56.9	59.1

of exemplar representations are sufficiently descriptive when tracking a single object.

4.5 CONCLUSION

We propose a novel Transformer layer for single-object visual tracking, based on Exemplar Attention. Exemplar Attention utilizes a single query token of the input sequence and jointly learns a small set of exemplar representations. The proposed transformer layer can be used throughout the architecture, *e.g.* as a substitute for a convolutional layer. Having a comparable computational complexity to standard convolutional layers while being more expressive, the proposed Exemplar Transformer layers can significantly improve the accuracy and robustness of tracking models with minimal impact on the model's overall runtime. E.T.Track, our Siamese tracker with Exemplar Transformer, significantly improves the performance compared to the convolutional baseline and other transformer variants. E.T.Track is capable of running in realtime on computationally limited devices such as standard CPUs.

5

FAST INTEREST POINT DETECTION, DESCRIPTION, AND MATCHING

Efficient detection and description of geometric regions in images is a prerequisite in visual systems for localization and mapping. Such systems still rely on traditional hand-crafted methods for efficient generation of lightweight descriptors, a common limitation of the more powerful neural network models that come with high compute and specific hardware requirements. In this chapter, we focus on the adaptations required by detection and description neural networks to enable their use in computationally limited platforms such as robots, mobile, and augmented reality devices. To that end, we investigate and adapt network quantization techniques to accelerate inference and enable its use on compute limited platforms. In addition, we revisit common practices in descriptor quantization and propose the use of a binary descriptor normalization layer, enabling the generation of distinctive binary descriptors with a constant number of ones. ZippyPoint, our efficient quantized network with binary descriptors, improves the network runtime speed, the descriptor matching speed, and the size of sparse 3D maps, by at least an order of magnitude when compared to full-precision counterparts. These improvements come at a minor performance degradation as evaluated on the tasks of homography estimation, visual localization, and map-free visual relocalization.

5.1 INTRODUCTION

The detection and description of geometric regions in images, such as salient points or lines, is one of the fundamental components in visual localization and mapping pipelines – essential prerequisites for AR and robotic applications. Achieving such detection and description efficiently with handcrafted algorithms [Rub+11; LCS11] has produced successful robot localization methods [MMT15; MT17; End+12; GT12; Leu+15]. On the other hand, DNNs have significantly advanced the representational capability of descriptors by learning on large scale natural images [DMR18], using deeper networks [Dus+19], or introducing new modules to learn

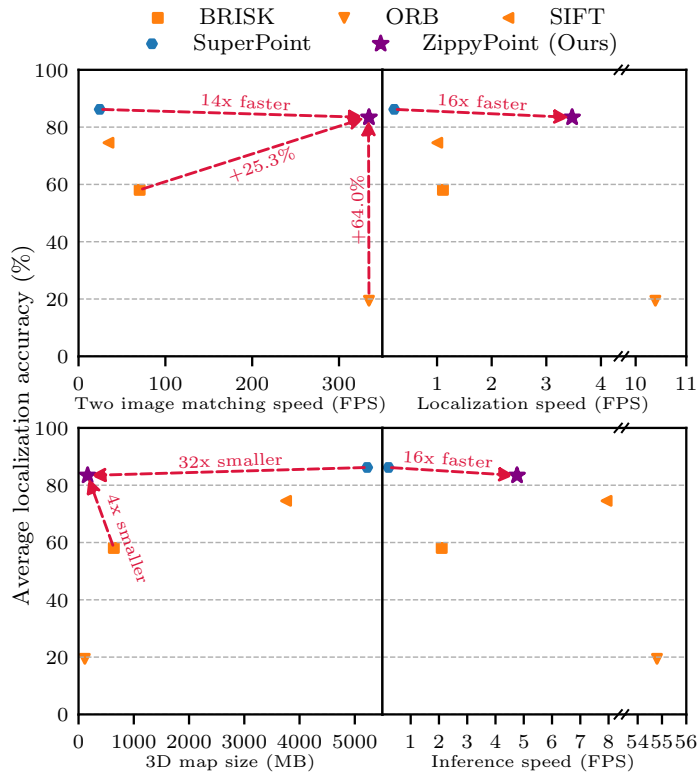


Figure 5.1: Learned detection and description methods, *e.g.* SuperPoint [DMR18], significantly outperform hand-crafted methods, in orange, on challenging day-night scenarios [Sat+18; Sat+12]. This, however, comes at the cost of slower image matching (top left), slower keypoint detection and description (bottom right), larger sparse 3D maps (bottom left), and therefore slow localization within a 3D space (top right). Speeds are reported in FPS on a CPU. In this chapter we present ZippyPoint, a learned detection and description network that improves the above limitations by at least an order of magnitude while providing competitive performance, enabling its use on-board computationally limited platforms.

feature matching [Sar+20]. However, these advances often come at the cost of more expensive models with slow run times and large memory requirements for representation storage, making them unsuitable for computationally limited platforms. While the demand for real-time applications such as robotics and AR is increasing, efficient DNN methods

that can operate in real-time on computationally limited platforms have received surprisingly little attention.

A key component for the successful deployment of mobile robots in large-scale applications is the real-time extraction of binary descriptors. This not only enables efficient storage of the detected representation, e.g. the map in Simultaneous Localization and Mapping (SLAM) or Structure-from-Motion (SfM) pipelines, but also accelerated descriptor matching. In particular, matching computations in localization scale non-linearly with the number of images or map size. Therefore, improved two-view matching speed can translate to very high gains in real applications. Fast and light weight descriptor methods include BRISK [LCS11], BRIEF [Cal+10] and ORB [Rub+11], however, their matching capability is often inferior to standard hand-crafted features such as SIFT [Low04] and SURF [BTG06], as presented by Heinly J. *et al.* [HDF12]. In challenging scenarios, however, hand-crafted feature extractors are outperformed significantly by learned representations [Sat+18]. While the performance gains of learned methods are highly desired, embedded platforms are limited in storage, memory, providing limited or no support for FP arithmetic, thus limiting the use of learned methods.

Motivated by the desire for improving the performance of feature points on low-compute platforms, we explore DNN quantization to enable the real-time generation of learned descriptors under such challenging constraints. However, the quantization of a DNN is not as straightforward as selecting the discretization level of convolutional layers. Quantized DNNs often require different levels of discretized precision for different layers [Ras+16; Liu+18b]. Operations such as max-pooling favour saturation regimes [Ras+16], while average pooling is affected by the required rounding and truncation operations. Moreover, prior works often focus on image-level classification tasks, with findings that do not necessarily transfer to new tasks [Bet+19]. To render the search for a Quantized Neural Network (QNN) tractable, we propose a *layer partitioning and traversal* strategy, a heuristic algorithm that divides the network components into blocks, and independently finds the optimal quantization configuration for each block, significantly reducing the architecture search complexity. While most research considers homogeneous quantization precision across all layers [Nag+19], we find Mixed-Precision (MP) quantization yields superior performance. In addition, we find that replacing standard pooling operations with learned alternatives can further improve QNN performance.

Besides the need for real-time inference, DNNs need to additionally generate binary local descriptors for storage efficiency and fast feature

matching. This adds further challenges as the discretization of the output layer draws less precise boundaries in the feature domain [LS20], making the network optimization more challenging. Furthermore, prior works focus on global feature description and present findings that do not trivially transfer to our task [She+18; Lai+15]. To this extent, we introduce a Binary Normalization (Bin.Norm) layer that constrains the representation to a constant pre-defined number of ones. Bin.Norm is therefore analogous to the L_2 normalization, a staple and key component in FP metric learning [MBL20].

In summary, our contributions are:

— We propose a heuristic algorithm, named *layer partitioning and traversal strategy*, to investigate the topological changes required for the quantization of a state-of-the-art detection and description network. We find that with a MP quantization architecture, and by replacing max-pooling with a learned alternative, we can achieve a speed-up by an order of magnitude with minor performance degradation. Our analysis reveals that the common QNN practices of using a single discretization level or standard pooling operations can be sub-optimal.

— We propose the use of a normalization layer for the end-to-end optimization of binary descriptors. Incorporating the Bin.Norm layer yields consistent improvements when compared to the common practices for descriptor binarization.

— We provide a detailed analysis of ZippyPoint, our proposed QNN with binary descriptors, on the task of homography estimation. We further demonstrate the generality of ZippyPoint on the challenging applications of Visual Localization (VisLoc) and Map-Free Visual Relocalization. ZippyPoint consistently outperforms all real-time alternatives and yields comparable performance to a full precision counterpart while addressing its known limitations of large sparse maps, slow image matching speed and slow network inference, illustrated in Fig. 5.1.

5.2 RELATED WORK

Hand-crafted feature extractors The design of hand-crafted sparse feature extractors such as SIFT [Low04] and SURF [BTG06] has been undoubtedly very successful in practice, still widely used in applications

such as SfM [SF16]. However, the time needed for detection and descriptor extraction, coupled with their FP representation, limits them from being used on compute-limited platforms, such as light-weight unmanned aerial vehicles. Motivated by this limitation, methods like BRISK [LCS11], BRIEF [Cal+10], and ORB [Rub+11], aimed to provide compact features targeted for real-time applications [MMT15; MT17; Leu+15]. While fast and lightweight, they lack the representational strength to perform well under a wide variety of viewing conditions such as large viewpoint changes [HDF12; SF16] or times of day and year [Sat+18].

Learned feature extractors Advances in DNNs have enabled the learning of robust, (pseudo-)invariant, and highly descriptive image features, pushing the boundaries of what was previously possible through hand-crafted methods. While hand-crafted local features [Low04; BTG06; LCS11; Cal+10; Rub+11; TLF09] have not evolved much, systematic incremental progress can be seen in the learned local features [Cho+16; DMR18; Fat+18; Ono+18; Rev+19; Tan+20; Dus+19]. Improvements have been achieved using contrastive learning [Cho+16], self-supervised learning [DMR18], improved architectures [Rev+19; Tan+20] and outlier rejection [Tan+20], to name a few approaches. Nevertheless, time and memory inefficiency remain major drawbacks of the learned methods.

In the same vein, large scale descriptor matching calls for light-weight representations. Binary descriptors enable efficient matching with moderate performance drops while significantly decreasing the storage requirements. Yet, the existing literature on binary representations focuses on image retrieval [Lin+16; She+18; Lai+15; She+15; Wan+17; NFS12], neglecting the detection and description of local features. For descriptor binarization, [She+18; She+15; Lin+16] rely on multiple distinct steps for the successful binarization of the descriptors. More similar to our work, [NFS12] defines a differentiable objective for the hamming distance, [Lai+15] uses sigmoids to soften the optimization objective, while [Tan+19] rely on a hard sign function and gradient approximations. In the same spirit, we also optimize the network in a single optimization step. However, we argue that the lack of normalization layer in these methods, a staple in metric learning [MBL20], greatly hinders the descriptor performance. To address this limitation, we propose a normalization layer for binary descriptors. Bin.Norm provides a more stable optimization process, avoids mode collapse and enables end-to-end optimization without requiring the use of gradient approximations or multiple optimization stages.

Efficient Neural Networks Several solutions have been proposed to deploy neural networks in constrained scenarios. These solutions can be partitioned in topological optimizations, aiming at increasing accuracy-per-operation or accuracy-per-parameter [HS15; How+19; Bla+23], software optimizations such as tensor decomposition and parameter pruning [Zha+15; Obu+20; Obu+21], and hardware-aware optimizations [Ras+16].

Amongst hardware-aware optimizations, quantization plays a central role [Ras+16; Liu+18b]. By replacing FP with Int operands, a QNN can reduce its storage and memory requirements with respect to an equivalent DNN. In addition, complex FP arithmetics can be replaced by simpler Int arithmetics. Due to these properties, QNNs can be executed at higher throughput (number of operations per cycle) and arithmetic intensity (number of arithmetic operations per memory transaction) [Jac18]. When operand precision is extremely low, e.g. Binary (Bin), standard instruction set architectures can be exploited to increase these metrics even further [Ras+16]. Unlike mainframes and workstations, embedded platforms have limited storage and memory, limited or no support for FP arithmetics, and are optimized to execute SIMD (Single Instruction, Multiple Data) Int arithmetics. These considerations make QNNs an ideal fit for embedded applications, such as robots and mobile devices.

However, QNNs have limited representational capacity compared to their FP counterparts. Specifically, linear operations using discretized weights draw less precise boundaries in their input domains. In addition, discretized activation functions lose injectivity with respect to their FP counterparts, making quantization a lossy process [LS20]. To strike a balance between throughput and performance, practitioners require to identify a single Int precision [Nag+19], or alternative linear layers [Liu+18b], that achieve the desired performance. These design choices are applied homogeneously across the entire network. We instead hypothesize that a single set of hyperparameters across the entire network can be suboptimal. We, therefore, investigate the use of heterogeneous layers throughout the network, e.g. different Int precision at different depths of the network, made possible through the proposed layer partitioning and traversal strategy.

5.3 MIXED PRECISION DISCRETIZATION

Efficiently identifying salient points in images and encoding them with lightweight descriptors is key to enabling real-time applications such as

robot localization. In this chapter, we explore the efficacy of learning-based descriptor methods under two constraints: minimizing run-time latency and using binary descriptors for accelerating keypoint matching and efficient storage. In Sec. 5.3.1 we introduce the baseline architecture we initiate our investigation from. In Sec. 5.3.2 we propose a strategy to explore structural changes to the network's topology. In Sec. 5.3.3 we introduce a standard formulation of metric learning, which we use to then define our Bin.Norm descriptor layer.

5.3.1 *Baseline Architecture*

We initiate our investigation from the state-of-the-art KP2D [Tan+20] network, which exploits outlier filtering to improve detections. The KP2D model maps an input image $\mathbf{I} \in \mathbb{R}^{H \times W \times 3}$ to keypoints $\mathbf{p} \in \mathbb{R}^{N \times 2}$, descriptors $\mathbf{x} \in \mathbb{R}^{N \times M}$, and keypoint scores $\mathbf{s} \in \mathbb{R}^{N}$, where N represents the total number of keypoints extracted and M the descriptor size. The model is comprised of an encoder with 4 VGG-style blocks [SZ15], followed by a three-headed decoder for keypoints, keypoint scores, and descriptors. The encoder is comprised of 8 convolutional operations, the keypoint and keypoint score branches of 2, and the descriptor branch of 4. All convolutional operations, except for the final layers, are followed by batch normalization and leaky ReLUs [MHN+13]. The model is optimized through self-supervision by enforcing consistency in predictions between a source image $\mathbf{I}_s$ and a target image $\mathbf{I}_t = \mathrm{H}(\mathbf{I}_s)$, related through a known homography transformation $\mathbf{H}$ and its warping function H.

We chose KP2D as the starting point for our investigation due to its standard architecture design choices: a VGG style encoder [DMR18; Chr+19; Dus+19; Rev+19], encoder-decoder structure [DMR18; Chr+19], and the detection and description paradigm [DMR18; Chr+19; Dus+19; Rev+19]. We expect that the investigated quantization strategy can be utilized by other similar models, such as the ones listed above, due to their architectural similarities.

5.3.2 *Network Quantization*

For the quantization of a convolutional layer, several design choices are required. These include weight precision, feature precision, and whether to use a high precision residual. When considering independently each layer of a DNN, it leads to a combinatorially large search grid, rendering an exhaustive search of the ideal quantization policy prohibitive.

To simplify the search space, we propose the *layer partitioning and traversal strategy*, depicted in Fig. 5.2. First, we partition the operations of our target architecture into macro-blocks. For each macro-block, we define a collection of candidate quantized configurations, such as weight precision. We then traverse through the macro-blocks and identify the optimal configuration for each, one at a time. This heuristic algorithm terminates once we have reached the most downstream network layer, the prediction heads. This strategy ensures that when a macro-block is optimized on features with given representation capabilities, it will not degrade due to optimization of a different macro-block upstream. Note that, while we maintain the macro-block configurations the same once selected, the architecture is always optimized end-to-end. By dividing the architecture into macro-blocks, we reduce the search complexity from combinatorial (the product of the number of configurations for each macro-block) for the greedy search, to linear (the sum of the number of configurations for each macro-block). We detail our choice of macro-blocks and their configurations in the experiments section.

5.3.3 *Binary Learned Descriptors*

Preliminaries When describing an image or a local region, the learned mapping aims to project a set of data points to an embedding space, where similar data are close together and dissimilar data are far apart. A fundamental component to the success of learned descriptors is the advancement of contrastive losses [HCL06; WS09; Den+19]. To ensure stable optimization and avoiding mode collapse, descriptors are often normalized [MBL20]. A common selection is L_2 normalization, defined as

$$\mathbf{y} = \frac{1}{||\mathbf{x}||_2}\mathbf{x}. \tag{5.1}$$

While the solution, and hence gradients, can be expressed in closed-form, it assumes FP representation spaces $\mathbf{x}$ and $\mathbf{y}$. However, Bin descriptors can only take discrete values {0,1}. In search for a normalization layer applicable for Bin descriptors, we instead view and rewrite Eq. (5.1) as the generalized optimization objective

$$\begin{aligned} \mathbf{y} = \underset{\mathbf{z}\in\mathbb{R}^M}{arg\,\min} \quad & d(\mathbf{z};\mathbf{x}) \\ \text{subject to} \quad & constr(\mathbf{z}), \end{aligned} \tag{5.2}$$

where we search for the vector $\mathbf{z}$ that minimizes a distance function to $\mathbf{x}$ under a normalization constraint $constr(\mathbf{z})$. Eq. (5.2) is therefore

equivalent to Eq. (5.1) when $d(\mathbf{z};\mathbf{x}) = \frac{1}{2}||\mathbf{z}-\mathbf{x}||_2^2$ and $constr(\mathbf{z})$ is $||\mathbf{z}||_2 = 1$. This enables the definition of optimization objectives that can be utilized where L_2 normalization does not provide the required behaviour. While constrained optimization problems are not differentiable, their use in DNNs is made possible through advances in deep declarative networks [GHC21].

Normalization for Binary Descriptors We hypothesize that normalization for binary descriptors is equivalent to having a constant number of ones in each descriptor. To this end, we take inspiration from multi-class classification problems [AKK19; MA16] and view the Bin.Norm as a projection of the descriptors living in an M-dimensional hypercube on a k-dimensional polytope [AKK19]. In other words, an M-dimensional descriptor has entries that sum to k. This trivially yields the constraint from Eq. (5.2) to $constr(\mathbf{z}) = \mathbf{1}^\top\mathbf{z} = k$, where $\mathbf{1}$ is a vector of 1s of the same dimension as $\mathbf{z}$.

We define the new optimization objective as

$$\begin{aligned}\mathbf{y} = \underset{\mathbf{z}\in[0,1]^M}{arg\,\min} & -\mathbf{x}^\top\mathbf{z} - H(\mathbf{z}) \\ \text{subject to} \quad & \mathbf{1}^\top\mathbf{z} = k\end{aligned} \tag{5.3}$$

where $H(\mathbf{z})$ is the binary entropy function applied on the vector $\mathbf{z}$ for entropy based regularization.

To optimize the objective, we introduce a dual variable $\nu \in \mathbb{R}$ for the constraint of Eq. (5.3). The Lagrangian then becomes

$$-\mathbf{x}^T\mathbf{z} - H(\mathbf{z}) + \nu(k - \mathbf{1}^\top\mathbf{z}). \tag{5.4}$$

Differentiating with respect to $\mathbf{z}$, and solving for first-order optimality gives

$$-\mathbf{x} + \log\frac{\mathbf{z}^*}{\mathbf{1}-\mathbf{z}^*} - \nu^* = 0, \tag{5.5}$$

after some manipulation

$$\begin{aligned} -\mathbf{x} - \nu^* &= -\log \frac{\mathbf{z}^*}{\mathbf{1} - \mathbf{z}^*} \\ -\mathbf{x} - \nu^* &= \log \frac{\mathbf{1} - \mathbf{z}^*}{\mathbf{z}^*} \\ e^{-(\mathbf{x}+\nu^*)} &= \frac{\mathbf{1} - \mathbf{z}^*}{\mathbf{z}^*} \\ \mathbf{z}^* e^{-(\mathbf{x}+\nu^*)} &= \mathbf{1} - \mathbf{z}^* \\ \mathbf{z}^* (1 + e^{-(\mathbf{x}+\nu^*)}) &= \mathbf{1} \\ \mathbf{z}^* &= \frac{\mathbf{1}}{\mathbf{1} + e^{-(\mathbf{x}+\nu^*)}} \end{aligned} \tag{5.6}$$

it yields

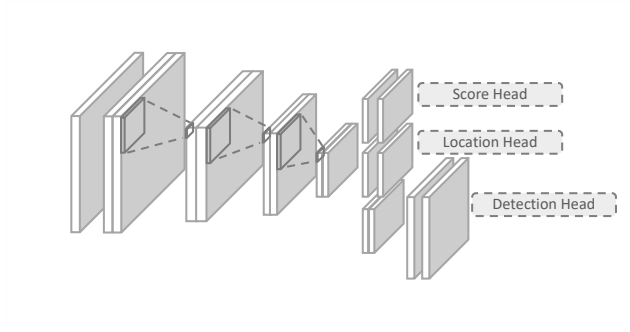
$$\mathbf{y} \approx \mathbf{z}^* = \sigma(\mathbf{x} + \nu^*), \tag{5.7}$$

where σ denotes the logistic function. We identify the optimal ν^* by using the bracketing method of [AKK19] that is efficiently implemented for use on GPUs, and backpropagate using [AK17].

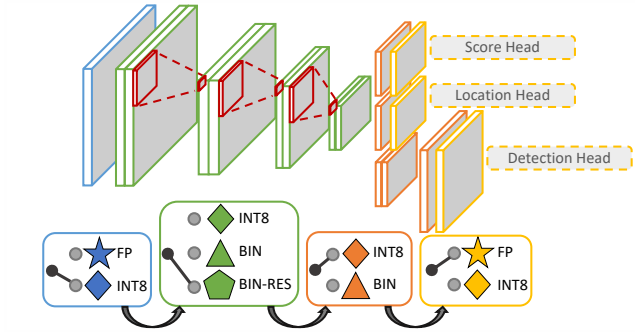
The selection of the optimization objective in Eq. (5.3) is two-fold. The entropy regularizer helps prevent sparsity in the gradients of the projection. In addition, the forward pass in Eq. (5.7) can be seen as an adaptive sigmoid that ensures the descriptor entries sum up to a specific value. This enables the direct comparison with the common practice of approximating binary entries using the sigmoid function.

5.4 EXPERIMENTS

We first present the implementation details in Sec. 5.4.1. In Sec. 5.4.2 we investigate the effect of network quantization using the layer partitioning and traversal strategy, as well as evaluate the proposed Bin.Norm layer for descriptor binarization. We then combine the two contributions into ZippyPoint and evaluate its performance on the task of homography estimation. We evaluate the generalization capabilities of ZippyPoint on fundamental tasks in robotic and AR pipelines, namely VisLoc in Sec. 5.4.3, and Map-Free Visual Relocalization in Sec. 5.4.4. Code and trained models will be released upon acceptance. We envision our work can both spark further research in the design of binary descriptors and quantized networks, as well as promote the incorporation of ZippyPoint in robotic systems.



(a) Detection and Description architecture



(b) The layer partitioning and traversal strategy

Figure 5.2: (a) Starting from [Tan+20], we partition the operations in macro-Blocks, depicted in (b) with different colors. From the first upstream macro-Block, in blue, we identify the optimal quantization setting that maintains functional performance while improving the network's throughput. We then traverse to the next block, green, and repeat. The strategy is complete once we have reached the most downstream network layer, the prediction heads.

5.4.1 *Implementation Details*

We implement our models in TensorFlow [Mar+15], and use the Larq [GT20] library for quantization. Our models are trained on the COCO 2017 dataset [Lin+14], comprised of 118k training images, following [DMR18; Chr+19; Tan+20]. The models are optimized using ADAM [KB15] for 50 epochs with a batch size of 8, starting with an initial learning rate of 10^{-3} while halving it every 10 epochs. To ensure robustness in our results, we optimize each model configuration three times and report the mean and standard deviation.

To enable self-supervised training, spatial and non-spatial augmentations for the homography transformation are required. For spatial

transformations, we utilize crop, translation, scale, rotation, and symmetric perspective. Non-spatial transformations applied are per-pixel Gaussian noise, Gaussian blur, color augmentation in brightness, contrast, saturation, and hue. Finally, we randomly shuffle the color channels and convert images to gray scale. Please refer to [Tan+20] for more details.

5.4.2 *Designing ZippyPoint*

We conduct our DNN quantization investigation on the task of homography estimation, a commonly used task for the evaluation of self-supervised learned models [DMR18; Chr+19; Tan+20]. Homographic transformations largely eliminates domain shifts due to the elimination of the third dimension, minimizing the adverse effects caused from known challenges such as occlusions, providing a good benchmark for ablation studies.

We evaluate our method on image sequences from the HPatches dataset [Bal+17]. HPatches contains 116 scenes, separated in 57 illumination and 59 viewpoint sequences. Each sequence is comprised of 6 images, with the first image used as a reference. The remaining images are used to form pairs for evaluation. As is common practice, we report Repeatability (Repeat.), Localization Error (Loc), Matching Score (M.Score), and Homography Accuracy with thresholds of 1, 3 and 5 pixels (Cor-1, Cor-3, Cor-5). We additionally benchmark and report the CPU speeds in Frames Per Seconds (FPS) on an Apple M1 ARM processor, since Larq currently only provides the optimized inference engine that exploits Bin convolutions for ARM processors.

Baseline We initiate our investigation in Table 5.1 from a re-implementation of KP2D with minor modifications to enable a structured search with minimal macro-block interference. Specifically, KP2D uses a shortcut connection between the encoder and decoder macro-blocks. We remove this skip-connection to constrain the interaction between two macro-blocks to a single point. Furthermore, we replace the leaky ReLUs with hard-swish [How+19], a comparable but faster alternative. The functional performance of *Baseline* is comparable to KP2D while slightly improving the throughput.

We then partition our baseline architecture into macro-blocks. These include the first encoder convolution, the remaining encoder convolutions, spatial reduction layers, the non-head decoder convolutions, and the head decoder convolutions, as depicted in Fig. 5.2 by the different colours.

Macro-Block I: First Encoder Convolution For macro-block I, we considered two configurations: FP and Int8. Although [Ras+16; Liu+18b]

suggest that keeping the first convolution in FP has a negligible effect on application throughput while degradation of functional performance, our findings suggest otherwise. Specifically, we find that using an Int8 convolution improves throughput by as much as 3 FPS, while having no detectable impact on functional performance. We ascribe this to the fact that the input images are also represented in Int8. Therefore, discretization of the input sequence does not cause a loss of information, while enabling the use of a more efficient Int8 convolution.

Macro-Block II: Encoder Convolutions For the encoder convolutions, we considered three configurations: Int8, Bin, and Binary with a high-precision Residual (Bin-R). While using Bin convolutions in the encoder significantly improves application throughput, functional performance is severely hindered, as measured by the halving of the correctness metrics. This drop is consistent with findings in the literature for semantic segmentation [Zhu+19] while conflicting with image-level classification experiments [Ras+16]. This further supports our arguments for the importance of task-specific investigations.

To alleviate such drastic performance drops, we introduce high precision Int8 representations in the form of a residual operation. For convolutional operations with a mismatch in the number of input and output channels, we introduce additional Int8 1×1 convolutions on the residual path. This ensures the high-precision paths maintain their Int8 precision, while matching the channel dimensions. The additional high-precision Int8 residuals improve performance significantly. This again advocates for the redundancy of FP representation in the encoder, as the encoder is now bottlenecked by Int8 precision.

Macro-Block III: Spatial Reduction For the spatial reduction layers, we considered four configurations: max-pooling (Max), average-pooling (Aver.), sub-sampling (Sub.S.) and a learned projection (Learn). As is common in DNNs, our baseline utilizes max-pooling. However, max-pooling has been found to favour saturated regimes and therefore eliminates information when applied on low-precision features like those found in QNNs [Ras+16]. Average pooling further degrades the performance, attributed to the errors introduced due to the roundings and truncations which are essential for integerized arrays. To further highlight this error, a simple Sub.S. that only uses information from a quarter of the kernel window yields comparable performance to Aver.

To alleviate the challenges highlighted above, we propose the use of a learned pooling operation (Learn). The learned pooling comes in the form of an Int8 convolutional operation with the same kernel size and stride

as the other pooling operations. We select Int8 so as to maintain the representational precision of the network, defined by the macro-block I and the high precision residuals. While operating at a comparable runtime to max pooling, the performance significantly improved. This further corroborates our hypothesis that learned pooling can address both the aforementioned challenges. Finally, we investigate the effect of the pooling placement (E.Learn). Specifically, we change the location of the pooling operations from the end to the beginning of each convolutional block. While with FP convolutional layers this would cause a 4× speedup for each convolution, in quantized convolutions the gain is even greater [GT20].

Macro-Block IV: Decoder Convolutions For the decoder convolutions, we considered two configurations: Int8, and Bin-R. We do not investigate Bin due to the large performance drop observed in macro-block II. Unlike the findings from macro-block II, our decoder experiments demonstrate the importance of Int8, highlighting the benefits of MP networks. Specifically, utilizing Int8 for the entire network would not yield the best throughput, as seen in macro-block II, while Bin-R for the entire network would not yield the best performance.

Macro-Block V: Final Decoder Convolutions For the final convolutions, we evaluated FP and Int8 for the score, location and descriptor heads independently. We find that score and location heads require FP representations, with models often failing to optimize otherwise. On the other hand, the descriptor branch can be optimized with Int8, significantly improving the throughput.

Network Quantization Findings Network latency can be significantly improved when quantizing the first convolutional layer and the last descriptor head to Int8, while having an insignificant effect on functional performance. This is contrary to findings from prior works that observed significant degradation in performance while claiming insignificant latency improvements [Ras+16; Liu+18b]. In addition, enhanced performance can be achieved through MP QNNs. In other words, a network comprised of only Bin-R or Int8 convolutional operations would yield sub-optimal results. This observation suggests that good quality and general-purpose features can be extracted using low-precision convolutions when coupled with higher precision residuals. Furthermore, it suggests that the dense task predictor heads benefit from higher Int8 precision to accurately reconstruct the target information from the encoded features. Finally, we observe that the prediction heads for regression tasks (score and location) cannot be quantized and should be left in FP, while the descriptor head

Table 5.1: Results from the layer partitioning and traversal strategy. The final model, in bold, performs comparably to the baseline while running an order of magnitude faster. Green highlights the configuration used in the next stage.

					Repeat. ↑	Loc. ↓	Cor-1 ↑	Cor-3 ↑	Cor-5 ↑	M.Score ↑	FPS ↑
KP2D [Tan+20]					0.686	0.890	0.591	0.867	0.912	0.544	2.5
Baseline (ours)					0.649 ± 0.004	0.792 ± 0.015	0.566 ± 0.015	0.880 ± 0.011	0.925 ± 0.007	0.571 ± 0.004	3.6
Encoder Convolutions											
First conv		Remaining convs									
FP	Int8	Int8	Bin	Bin-R							
✓		✓			0.651 ± 0.004	0.840 ± 0.076	0.528 ± 0.019	0.867 ± 0.017	0.922 ± 0.011	0.574 ± 0.004	10.6
	✓	✓			0.657 ± 0.003	0.825 ± 0.015	0.548 ± 0.003	0.866 ± 0.004	0.925 ± 0.010	0.577 ± 0.001	13.8
	✓		✓		0.561 ± 0.036	1.120 ± 0.061	0.281 ± 0.132	0.401 ± 0.022	0.449 ± 0.080	0.247 ± 0.096	15.2
	✓			✓	0.653 ± 0.005	1.040 ± 0.018	0.401 ± 0.034	0.811 ± 0.011	0.890 ± 0.007	0.563 ± 0.006	14.5
Spatial Reduction											
Max	Aver.	Sub.S.	Learn	E.Learn							
✓					0.653 ± 0.005	1.040 ± 0.018	0.401 ± 0.034	0.811 ± 0.011	0.890 ± 0.007	0.563 ± 0.006	14.5
	✓				0.656 ± 0.003	1.068 ± 0.033	0.354 ± 0.038	0.788 ± 0.027	0.873 ± 0.011	0.558 ± 0.015	13.9
		✓			0.640 ± 0.022	1.128 ± 0.053	0.362 ± 0.024	0.783 ± 0.003	0.881 ± 0.007	0.537 ± 0.015	14.4
			✓		0.648 ± 0.009	0.890 ± 0.066	0.517 ± 0.029	0.848 ± 0.009	0.916 ± 0.003	0.571 ± 0.006	14.2
				✓	0.656 ± 0.002	0.943 ± 0.034	0.491 ± 0.024	0.844 ± 0.015	0.906 ± 0.005	0.568 ± 0.002	16.2
Decoder Convolutions											
Remaining convs			Descriptor conv								
Int8	BIN-R		FP	Int8							
✓			✓		0.658 ± 0.002	0.964 ± 0.020	0.488 ± 0.030	0.840 ± 0.013	0.900 ± 0.001	0.569 ± 0.002	27.2
	✓		✓		0.655 ± 0.003	1.018 ± 0.006	0.451 ± 0.001	0.456 ± 0.001	0.481 ± 0.002	0.329 ± 0.008	30.1
✓				✓	**0.652 ± 0.005**	**0.926 ± 0.022**	**0.506 ± 0.025**	**0.853 ± 0.007**	**0.917 ± 0.003**	**0.571 ± 0.003**	**47.2**

can be quantized to Int8. This further drives the importance of the structured investigation, like the layer partitioning and traversal strategy.

Binarizing descriptors We initiate the descriptor exploration from the common practice of utilizing sigmoid as a soft approximation for every bit [Lai+15; Liu+12], and the hamming triplet loss proposed by Lai et al. [Lai+15]. While some works use a hard sign function [Tan+19], we found it unable to optimizate the network to a meaningful degree. Table 5.2 demonstrates a significant performance drop and large variance compared to the baseline, especially in the correctness metrics. We conjecture that this spans from the saturation of the sigmoids, yielding uninformative gradients. We hypothesize that the aforementioned limitation can be partly addressed by the use of a normalization layer, and test this assumption by appending an $||L_2||$ normalization layer after the element-wise sigmoid operation. This constrains the activations and dramatically improves performance and reduces the variance, as seen experimentally, leading to a more stable optimization process.

In this chapter, we hypothesize that analogous to $||L_2||$ normalization, we can optimize the network using a Bin normalization layer by constraining the descriptor to a constant number of ones. Using the proposed Bin.Norm layer, the functional performance gap is significantly decreased when compared to the FP descriptors. Note that, during inference the binary descriptor optimization strategy in Eq. (5.3) can be replaced by a thresholding function that sets the top-k logits of each descriptor to 1, enabling faster processing.

Comparison to state-of-the-art We compare ZippyPoint with state-of-the-art methods in Table 5.3. For a fair comparison, we group methods given the descriptor precision. When utilizing FP descriptors, ZippyPoint performs on par with other methods. In particular, it consistently outperforms SuperPoint and performs on par with KP2D. Meanwhile, the throughput gain is higher than an order of magnitude.

The benefits of ZippyPoint, the combination of the fast QNN architecture from Table 5.1 and the binary optimization strategy from Table 5.2, become apparent when comparing binary descriptor methods. We consistently outperform ORB [MMT15] by a large margin in all metrics. We additionally outperform BRISK [LCS11] in all metrics and even report double the matching score, a crucial metric for adaptation of these methods in downstream tasks like VisLoc. Here on, we refer to ZippyPoint as our QNN with binary descriptors.

Table 5.2: We evaluate the efficacy of different normalization layers when combined with using sigmoid as a soft approximation for every bit. We find that the binary normalization (Bin.Norm) layer for the descriptors consistently improves all metrics.

	Norm	Repeat. ↑	Loc. ↓	Cor-1 ↑	Cor-3 ↑	Cor-5 ↑	M.Score ↑
Full Precision	L_2	0.644 ± 0.003	0.788 ± 0.005	0.580 ± 0.007	0.886 ± 0.008	0.933 ± 0.010	0.569 ± 0.003
Sigmoid		0.640 ± 0.005	0.809 ± 0.049	0.173 ± 0.300	0.285 ± 0.493	0.305 ± 0.528	0.187 ± 0.318
Sigmoid + $\|\|L_2\|\|$	L_2	0.650 ± 0.001	0.803 ± 0.010	0.491 ± 0.015	0.822 ± 0.009	0.888 ± 0.003	0.513 ± 0.003
Sigmoid + Bin.Norm (Ours)	Bin.Norm	**0.651 ± 0.003**	**0.796 ± 0.016**	**0.545 ± 0.005**	**0.880 ± 0.090**	**0.925 ± 0.004**	**0.553 ± 0.002**

Table 5.3: We compare ZippyPoint with full precision or binary descriptors against state-of-the-art methods. ZippyPoint performs on par with other full-precision methods while running an order of magnitude faster than the full-precision alternative. When compared to binary hand-crafted methods, ZippyPoint consistently outperforms all other methods, often by a large margin.

	Repeat. ↑	Loc. ↓	Cor-1 ↑	Cor-3 ↑	Cor-5 ↑	M.Score ↑
			Full-Precision Descriptors			
SuperPoint [DMR18]	0.631	1.109	0.491	0.833	0.893	0.318
SIFT [Low04]	0.451	**0.855**	**0.622**	0.845	0.878	0.304
SURF [BTG06]	0.491	1.150	0.397	0.702	0.762	0.255
KP2D [Tan+20]	**0.686**	0.890	0.591	**0.867**	0.912	0.544
ZippyPoint (Ours)	0.652 ± 0.005	0.926 ± 0.022	0.506 ± 0.025	0.853 ± 0.007	**0.917 ± 0.003**	**0.571 ± 0.003**
			Binary Descriptors			
BRISK [LCS11]	0.566	1.077	0.414	0.767	0.826	0.258
ORB [Rub+11]	0.532	1.429	0.131	0.422	0.540	0.218
ZippyPoint (Ours)	**0.652 ± 0.005**	**0.926 ± 0.022**	**0.433 ± 0.007**	**0.820 ± 0.007**	**0.887 ± 0.006**	**0.571 ± 0.003**

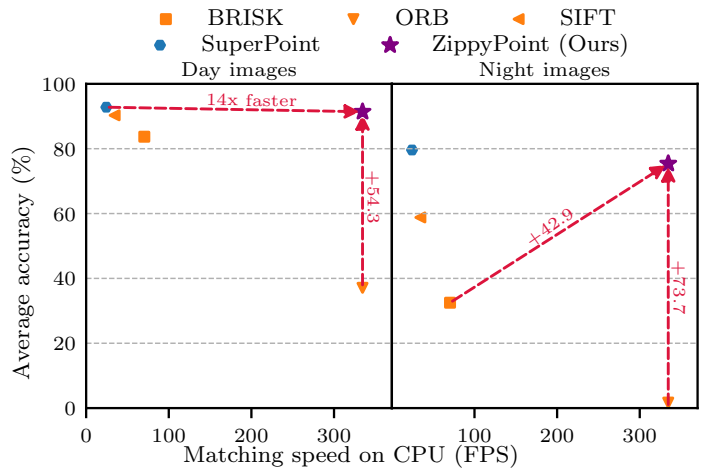


Figure 5.3: Comparison of the average visual localization accuracy vs descriptor matching speed between two images on the AachenV1.1 Day-Night datasets. ZippyPoint consistently outperforms all other binary methods.

5.4.3 *Visual Localization*

Camera localization is one of the key components in several robotic and mapping applications. Both relative [Nis04] and absolute [KSS11] camera localization require good local feature point descriptors to match, and are key building blocks in seminal pipelines [MMT15; MT17; End+12; GT12; Leu+15; Dav+07]. To further demonstrate the potential of ZippyPoint, we assess its generalization capability on the task of absolute camera localization, where the pose of a query image is estimated with respect to a 3D map.

We utilize the hloc framework [Sar+19], similar to prior works [Rev+19; Sar+20], and evaluate the performance on the challenging real-life Aachen V1.1 Day-Night datasets from the VisLoc benchmark [Sat+18; Sat+12]. More precisely, we reconstruct the 3D map using ZippyPoint features instead of SIFT [Low04]. For each query image, we perform a coarse search of the map and retrieve the 30 closest database images based on their global descriptors, representing candidate locations. The query image is then localized within the 3D map by utilizing the candidate locations. Please refer to [Sar+19] for more details.

Table 5.4 presents the performance breakdown for both the day and night datasets. In addition, we report the size of the 3D map (Map)

Table 5.4: Comparison of the visual localization accuracy, given different error threshold, on the AachenV1.1 Day-Night datasets. We additionally report the size of the sparse 3D map (Map), the localization speed (Loc.) for descriptor extraction and matching in the hloc framework [Sar+19], the inference speed for the extraction of the descriptors (Inf.), and the matching speed for two images (Match.). The arrows indicate the improvement direction. ZippyPoint consistently outperforms all other binary descriptor methods, while yielding great trade-offs with respect to inference speed, matching speed, and model size.

	Day ↑			Night ↑			Map (MB) ↓	Loc. (FPS) ↑	Inf. (FPS) ↑	Match. (FPS) ↑
m	0.25	0.50	5.00	0.50	1.00	5.00				
deg	2	5	10	2	5	10				
Full-Precision Descriptors										
SuperPoint [DMR18]	86.8	93.8	97.9	62.3	81.7	94.8	5224	0.22	0.29	24.4
SIFT [Low04]	82.3	91.6	97.0	45.0	58.6	72.8	3756	1.00	7.93	34.5
Binary Descriptors										
BRISK [LCS11]	75.2	84.1	92.4	23.0	32.5	41.9	638	1.11	2.10	70.4
ORB [Rub+11]	25.4	35.3	50.6	1.0	1.6	2.6	113	10.39	54.80	334.5
ZippyPoint (Ours)	**85.0**	**92.2**	**97.0**	**63.4**	**74.9**	**88.0**	163	3.47	4.76	334.5

in megabytes, the localization speed (Loc.) for descriptor extraction and matching in the hloc framework [Sar+19], the inference speed for the extraction of the descriptors (Inf.), and the matching speed for two images (Match.). These results are additionally depicted Fig. 5.1, by reporting the average performance score for both day and night query sets. Furthermore, the day and night results are presented in Fig. 5.3 with respect to the FPS speed for matching two images. Note that, the inference speed reported in Table 5.4 is lower than that of Table 5.1. This is attributed to the fact that the inference speed for learned methods scales linearly with the number of spatial dimensions in the input image. The image resolution used in Table 5.1 was 240×320, following [Tan+20], while in Table 5.4 the largest image dimension was rescaled to 1020, following [Sar+19].

While ZippyPoint performs comparably to SuperPoint during day time, we decrease the 3D map size, query localization time, and model inference speed by at least an order of magnitude. This is attributed to the lightweight binary descriptors, the more efficient similarity comparison between the descriptors, and the network quantization. Localization with ZippyPoint at night is slightly inferior to SuperPoint, however, we expect optimization of the image transformations during training can close this gap further.

On the binary descriptor front, ZippyPoint consistently outperforms ORB by a significant margin at a comparable matching speed. BRISK on the other hand is competitive to ours on the day dataset, with the slower run-time of BRISK attributed partly to the larger descriptor size, twice that of ZippyPoint, and the increased number of detected keypoints. However, the more challenging night dataset paints a different picture, with ZippyPoint outperforming BRISK by 42.9% and ORB failing to localize. This further attests to the need for efficient learned detection and description networks, in particular for more challenging and adverse conditions.

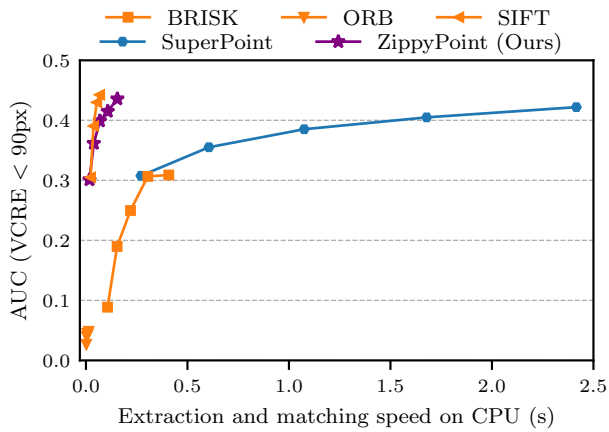
5.4.4 *Map-free Visual Relocalization*

Absolute camera localization, such as the task presented in Sec. 5.4.3, require an accurate 3D scene-specific map. This entails hundreds of images and large storage space, prerequisites that do not often hold in AR applications. These limitations have given rise to the more challenging Map-free Visual Relocalization benchmark [Arn+22]. The aim of Map-free Visual Relocalization is to predict the metric pose of a query image

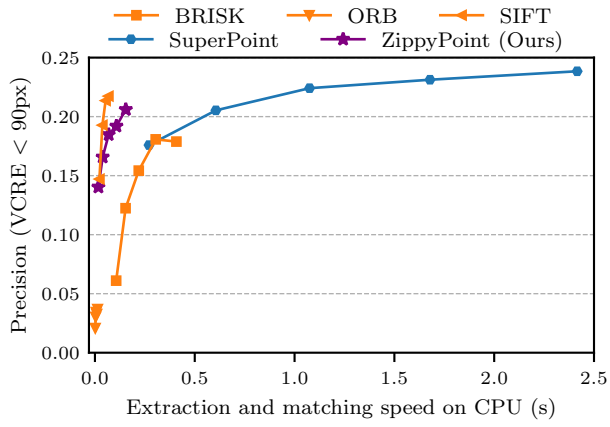
with respect to a single reference image that is considered representative of the scene of interest.

We evaluate interest point detection and description networks on the challenging Map-free Visual Relocalization benchmark. Specifically, as in [Arn+22], we first compute the Essential matrix [HZ03] between the query and the reference image using the 5-point solver [Nis04] of [Bar+20]. We then recover the scale using the estimated depth generated from a DPT model [RBK21] that has been fine-tuned on the KITTI dataset [GLU12]. We report the Area Under the Curve (AUC) and precision for pose error (Err) under the threshold of 25cm and 5-degree. In addition, we report AUC and Err for Virtual Correspondence Reprojection Error (VCRE) at an offset threshold of 10%, 90 pixels, simulating the placement of AR content in the scene [Arn+22]. The performances are reported in Fig. 5.4, Fig. 5.5, and Table 5.5 with respect to the latency for keypoint extraction and matching. For Fig. 5.4 and Fig. 5.5, we identify Pareto curves by rescaling the input images at ratios of 0.4 to 1.0 in 0.2 increments, a common practice to accelerate inference post-training, and also increase the ratio to 1.2 in order to evaluate if performance can improve further, as commonly done in VisLoc [Sar+19]. We also investigated larger ratios but found they often degraded the performance of the hand-crafted methods, such as SIFT, while the performance quickly plateaued for the DNNs.

We find that ZippyPoint yields comparable performance to SuperPoint while being an order of magnitude faster for feature extraction and matching. Additionally, ZippyPoint consistently outperforms the binary methods, BRISK and ORB, by a large margin. When compared to SIFT [Low04], however, ZippyPoint yields comparable results at a slight increase in latency. This is attributed to the nature of the dataset and task. Specifically, the Map-free Visual Relocalization benchmark presents a wide baseline benchmark without challenging long-term changes, the scenario under which SIFT shines. We expect similar benchmarks with long-term changes, similar to VisLoc, would better showcase the benefits of ZippyPoint, and the learned methods in general. Furthermore, while SIFT's keypoint matching is slower than ZippyPoint's, matching only takes place between a single pair of images for each scene in this experiment and therefore does not aggregate to a significantly large delay, unlike in VisLoc and SLAM where matching speed is often the bottleneck due to the required matching within a large map.

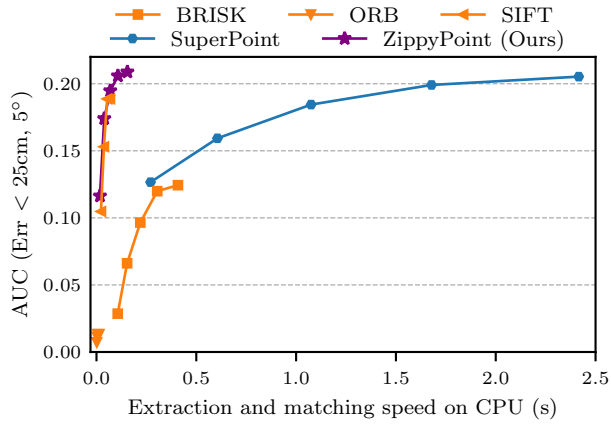


(a) Virtual Correspondence Reprojection Error AUC.

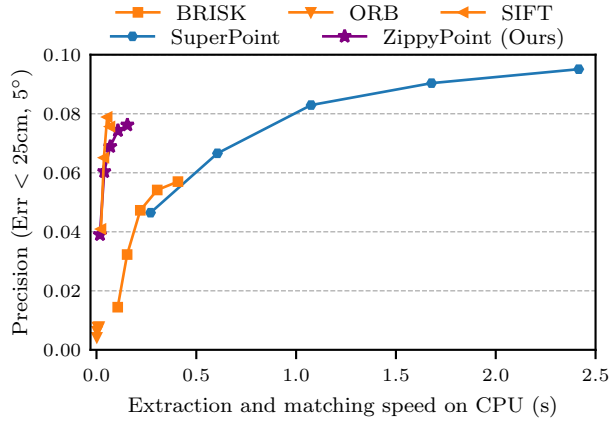


(b) Virtual Correspondence Reprojection Error precision.

Figure 5.4: Comparison of the different detection and description networks on the Map-free Visual Relocalization benchmark [Arn+22]. We report the Area Under the Curve (AUC) and precision under the Virtual Correspondence Reprojection Error (VCRE) with respect to the feature extraction and image matching speed. ZippyPoint consistently outperforms all binary descriptor methods and achieves comparable performance to SuperPoint at a significant speedup.



(a) Pose error AUC.



(b) Pose error precision.

Figure 5.5: Comparison of the different detection and description networks on the Map-free Visual Relocalization benchmark [Arn+22]. We report the Area Under the Curve (AUC) and precision under the pose error (Err) with respect to the feature extraction and image matching speed. ZippyPoint consistently outperforms all binary descriptor methods and achieves comparable performance to SuperPoint at a significant speedup.

Table 5.5: Comparison of the different detection and description networks on the Map-free Visual Relocalization benchmark [Arn+22] at the original image resolution. We report the Area Under the Curve (AUC) and precision under the Virtual Correspondence Reprojection Error (VCRE) and pose error (Err) with respect to the feature extraction and image matching speed (Latency) in seconds (s). ZippyPoint yields comparable performance to SuperPoint while being an order of magnitude faster. Additionally, ZippyPoint consistently outperforms the binary methods, BRISK and ORB, by a large margin.

	AUC ↑		Precision ↑		Latency (s) ↓
	VCRE < 90px	Err < 25cm, 5deg	VCRE < 90px	Err < 25cm, 5deg	
Full-Precision Descriptors					
SuperPoint [DMR18]	0.405	0.199	0.231	0.090	1.678
SIFT [Low04]	0.443	0.189	0.217	0.076	0.068
Binary Descriptors					
BRISK [LCS11]	0.307	0.120	0.181	0.054	0.304
ORB [Rub+11]	0.044	0.013	0.033	0.007	0.009
ZippyPoint (Ours)	0.415	0.206	0.192	0.074	0.107

5.5 CONCLUSION

In this chapter, we investigated efficient detection and description of learned local image points through mixed-precision quantization of network components and binarization of descriptors. To that end, we followed a structured investigation, we refer to as layer partitioning and traversal for the quantization of the network. In addition, we proposed the use of a binary normalization layer to generate binary descriptors with a constant number of ones.

We obtained an order of magnitude throughput improvement with minor degradation of performance. In addition, we find that the binary normalization layer allows the network to operate on par with full-precision networks, while consistently outperforming hand-crafted binary descriptor methods. The results show the suitability of our approach on visual localization and map-free visual relocalization, challenging downstream tasks and essential prerequisites for robotic applications, while significantly decreasing the 3D model size, matching, and localization speed. We believe ZippyPoint can spark further research towards bringing learned binary descriptor methods to mobile platforms, as well as promote its incorporation in both new and established robotic pipelines.

6

DISCUSSION

6.1 SUMMARY OF CONTRIBUTIONS

In this thesis, we discussed the importance of efficient DNNs and their impact in the fields of robotics, AR, and self-driving cars, amongst others. We presented four works that aim to increase the capabilities of DNNs while respecting common limitations of computationally constrained devices. Specifically, we show that (a) new dense prediction tasks can be incrementally learned with only a moderate increase in the total number of parameters. (b) Self-supervised tasks can be used as auxiliary tasks in MTL to increase performance and robustness, while also eliminating the need for additional labeling efforts. (c) Designing efficient transformer modules can significantly improve performance without compromising latency. Finally, (d) mixed precision discretization of DNNs and binarization of descriptors can improve efficiency without compromising performance.

In Chapter 2 we presented RCM, a method to reparameterize convolutions for MTL. By reparameterizing convolutional operations of standard neural network architectures into a non-trainable shared filter bank and task-specific modulators, we address two main challenges of MTL, namely, incremental learning and task interference. We evaluate RCM on two challenging benchmarks, PASCAL-Context [Mot+14] and NYUD [Sil+12], and find RCM can achieve comparable results to single-task models at a significant decrease in total parameters. Furthermore, we find RCM outperforms the standard MTL practice, as well as other state-of-the-art task-conditional MTL models.

In Chapter 3 we introduced CompL, a method that exploits the inductive bias provided by a self-supervised task to enhance the performance and robustness of a target task for an identical architecture. CompL exploits the label-free supervision of self-supervised methods, facilitating faster iterations through different task combinations. We show consistent performance improvements in fully and partially labeled datasets for both semantic segmentation and monocular depth estimation. While our method eliminated the need for labeling the auxiliary task, it commonly

outperforms the traditional MTL practice with labeled auxiliary tasks on monocular depth estimation. Additionally, the semantic segmentation models trained under CompL yield better robustness to zero-shot cross-dataset transfer.

In Chapter 4 we presented Exemplar Transformer, a novel transformer layer for single-object visual tracking based on Exemplar Attention that can improve performance at an insignificant increase in model runtime. Exemplar Attention utilizes a single query token of the input sequence and jointly learns a small set of exemplar representations. The proposed transformer layer can be used throughout the architecture, *e.g.* as a substitute for a convolutional layer. Having a comparable computational complexity to standard convolutional layers while being more expressive, the proposed Exemplar Transformer layers can significantly improve the accuracy and robustness of tracking models with minimal impact on the model's overall runtime. E.T.Track, our Siamese tracker with Exemplar Transformer, significantly improves the performance compared to the convolutional baseline and other transformer variants. E.T.Track is capable of running in realtime on computationally limited devices such as standard CPUs.

Finally, in Chapter 5 we investigated efficient detection and description of learned local image points through mixed-precision quantization of network components and binarization of descriptors. We presented a structured investigation, which we refer to as layer partitioning and traversal, for the quantization of the network. In addition, we proposed the use of a binary normalization layer to generate binary descriptors with a constant number of ones. We obtained an order of magnitude throughput improvement with minor degradation of performance. Furthermore, we find that the binary normalization layer allows the network to operate on par with full-precision networks, while consistently outperforming hand-crafted binary descriptor methods. The results show the suitability of our approach on visual localization and map-free visual relocalization, challenging downstream tasks and essential prerequisites for robotic applications, while significantly decreasing the 3D model size, matching, and localization speeds.

6.2 DISCUSSION, LIMITATIONS, AND FUTURE RESEARCH

This section discusses numerous limitations of the contributions presented in this thesis and aims to highlight directions for future research.

6.2.1 *Reparameterizing Convolutions for Multi-Task Learning*

Incremental task, but not class RCM enables the easy addition of new tasks, simply by optimizing new task-specific modulators. However, adding new classes in an already optimized task, *e.g.* adding the class cat on the task of semantic segmentation requires the retraining of the task-specific parameters. We expect that RCM can be easily combined with existing works on incremental class learning [Fre99; Reb+17; Kir+17; Lee+17; LH17; Wu+19], however, this has not been evaluated. This direction opens up the possibility of joint research in incremental task and class learning.

Elimination of task interactions is not always desired While jointly optimizing a large number of tasks can lead to performance degradation, jointly optimizing related tasks can lead to performance improvements [LDJ19; Sta+20; Bru+20]. This opens up opportunities for identifying task pairs that can be jointly optimized so as to both improve performance and decrease further the total number of parameters.

Improved filter bank In Chapter 2 we found that an ImageNet-optimized filter bank is sufficiently descriptive to enable comparable performance to single-task models. We hypothesize that the advancements in self-supervised learning [He+20], and in particular those acting on the local level [Wan+21d; Hen20], open up opportunities to learn more general-purpose features for dense tasks, and therefore improve performance further.

RCM was not designed for separable convolutions Efficient neural networks used on edge devices often make use of separable convolutions [JVZ14; How+17; San+18; TL19], however, RCM was evaluated on architectures using the standard convolutional operations. We hypothesize that both methods can be utilized in conjunction, however, this needs to be evaluated to better understand the tradeoffs in performance and parameters.

6.2.2 *Composite Learning for Dense Predictions*

Utilize CompL for state-of-the-art improvements In Chapter 3 we conducted a systematic study of CompL on a number of target tasks, such as semantic segmentation and monocular depth estimation. For this study, we follow the standard practice of MTL research and utilized

generic settings such as architectures and augmentations. Future research can focus on utilizing CompL in conjunction with state-of-the-art target task models to further improve benchmark performance. Furthermore, additional target tasks can be evaluated, such as human pose estimation [NYD16; Cao+17; Sun+19; Cao+19], object detection [Gir+14; Gir15; Ren+15; Red+16], and visual object tracking [Ber+16; Bha+19].

CompL self-supervised task design In Chapter 3 we focused on an extensive evaluation to demonstrate and motivate the benefits of jointly optimizing a supervised target task with existing self-supervised auxiliary tasks. Self-supervised tasks for pre-training often aim to learn image-level semantics, such as MoCo, however, tasks like boundary detection require self-supervised tasks that further enforce the meaning of a boundary. Such knowledge can be used to better guide the design of target task specific self-supervised tasks for joint training. Future research can therefore focus on practices that can enable the efficient design of novel self-supervised tasks for joint training with target tasks. Furthermore, the automation of these practices would be ideal if CompL is to be utilized out of the box for any target task since no single auxiliary task can improve all target tasks, as demonstrated in MTL literature [Sta+20].

6.2.3 *Efficient Visual Tracking with Exemplar Transformers*

Optimization challenges Throughout our experiment section, our E.T.Track utilized 4 exemplars. While increasing the number of exemplars should also increase the performance due to increased network capacity, similar to increasing a convolutional layer's width, we found the performance degraded. We attribute this performance degradation to optimization challenges, specifically, (a) insufficient optimization for all exemplars, and (b) inefficient use of the additional capacity. We expect optimization tricks, such as the stochastic dropping of exemplars, would enforce a more uniform update of the exemplars and minimize the strong reliance on a small subset of the available exemplars. Similar optimization tricks have been found to work for neuron dropout in fully connected layers [Sri+14], as well as entire modules along the network's depth [Hua+16].

Evaluate the efficacy of Exemplar Transformers on other tasks Chapter 4 focused on the design of an efficient single instance-level attention layer for realtime visual object tracking. We hypothesize that our efficient transformer layer can be beneficial to other tasks that require

the detection of a single instance, and can therefore assist in improving the state-of-the-art of such methods further. This includes, but is not limited to, image-level tasks such as image classification, dense prediction tasks like visual saliency segmentation [Ull+20], or even in specific parts of the network, such as the mask head in instance segmentation architectures [He+17].

6.2.4 *Fast Interest Point Detection, Description, and Matching*

Improve inference speed further ZippyPoint, our efficient QNN with binary descriptors, was built on the work of Tang et al. [Tan+20]. While inference speed has been improved by an order of magnitude, the latency might still be prohibitive for some applications. We recommend expanding our investigation by commencing the layer partitioning and traversal strategy from more efficient architectures [How+17; San+18; TL19]. Efficient architectures, however, rely on depthwise separable convolutions, whose appropriate method of quantization is still underexplored in the literature and could therefore cause challenges during network optimization.

Incorporate ZippyPoint in robotic and AR pipelines We have demonstrated the superior performance of ZippyPoint when compared to binary hand-crafted alternatives on three tasks, namely, homography estimation, visual localization, and map-free visual relocalization. We envision ZippyPoint can be used as a core building block in both new and established AR and robotic pipelines, amongst others. However, this requires substantial engineering effort on the pipeline front since existing pipelines are often overengineered for the descriptors at hand, and therefore the utilization of ZippyPoint is not as simple as plug and play.

6.3 OPEN-SOURCED CONTRIBUTIONS

One of the reasons the fields of machine learning and computer vision are advancing at a very high pace is that, alongside most papers, their corresponding codebases are released to the public. This ensures reproducible research and allows future works to make use of, and build, upon existing works. Inspired by the community's transparency, we open-source accompanying codebases for every publication presented in this thesis to help advance research further. Specifically:

— All resources for the paper "Reparameterizing Convolutions for Incremental Multi-Task Learning without Task Interference" (Chapter 2) are publicly available at https://github.com/menelaoskanakis/RCM

— All resources for the paper "Composite Learning for Robust and Effective Dense Predictions" (Chapter 3) are available at https://github.com/menelaoskanakis/CompL

— The model and inference code for the paper "Efficient Visual Tracking with Exemplar Transformers" (Chapter 4) are publicly available at https://github.com/pblatter/ettrack

— The model and accompanying demo for the paper "ZippyPoint: Fast Interest Point Detection, Description, and Matching through Mixed Precision Discretization" (Chapter 5) are available at https://github.com/menelaoskanakis/ZippyPoint

6.4 BROADER IMPACT STATEMENT

The majority of this thesis is focused on the design of efficient neural networks, however, it is also as important to reflect on the potential societal impact the conducted research can have. More efficient deep neural networks can enable realtime applications, such as robots and AR applications, to utilize more powerful models and therefore significantly improving their capabilities. This is highly desirable in a large number of applications, such as healthcare robots and AR guided surgeries, where mistakes can cause people to lose their lives. However, just like any other technological advancement, there is a risk of such models being used maliciously. This can include applications such as unmanned aerial vehicles for military purposes, or mass surveillance system to illegally track and locate individuals. Nevertheless, such negative impact is primarily associated to the application, rather than the specific technology and content presented in this thesis. We believe that proper legislation need to take into consideration the harmful use cases, and through strict supervision, ensure the elimination of such applications while still enabling their use in scenarios where they can help advance humanity.

BIBLIOGRAPHY

[AK17] Brandon Amos and J Zico Kolter. „Optnet: Differentiable optimization as a layer in neural networks.“ 2017. [↰ 84]

[AKK19] Brandon Amos, Vladlen Koltun, and J Zico Kolter. „The limited multi-label projection layer.“ 2019. [↰ 83, 84]

[Arb+10] Pablo Arbelaez, Michael Maire, Charless Fowlkes, et al. „Contour detection and hierarchical image segmentation.“ 2010. [↰ 47, 48]

[Arn+22] Eduardo Arnold, Jamie Wynn, Sara Vicente, et al. „Map-Free Visual Relocalization: Metric Pose Relative to a Single Image.“ 2022. [↰ 94–98]

[Bal+17] Vassileios Balntas, Karel Lenc, Andrea Vedaldi, et al. „HPatches: A benchmark and evaluation of handcrafted and learned local descriptors.“ 2017. [↰ 86]

[Bal81] Dana H Ballard. „Generalizing the Hough transform to detect arbitrary shapes.“ 1981. [↰ 1]

[Bar+20] Daniel Barath, Jana Noskova, Maksym Ivashechkin, et al. „MAGSAC++, a fast, reliable and accurate robust estimator.“ 2020. [↰ 95]

[Bax00] Jonathan Baxter. „A model of inductive bias learning.“ 2000. [↰ 31, 40]

[Bel+19] Irwan Bello, Barret Zoph, Ashish Vaswani, et al. „Attention augmented convolutional networks.“ 2019. [↰ 52, 54]

[Ber+16] Luca Bertinetto, Jack Valmadre, Joao F Henriques, et al. „Fully-convolutional siamese networks for object tracking.“ 2016. [↰ 2, 51, 53, 65, 66, 104]

[Bet+19] Joseph Bethge, Haojin Yang, Marvin Bornstein, et al. „Back to Simplicity: How to Train Accurate BNNs from Scratch?“ 2019. [↰ 77]

[Bey+22] Lucas Beyer, Xiaohua Zhai, Amélie Royer, et al. „Knowledge distillation: A good teacher is patient and consistent.“ 2022. [↰ 4]

[Bha+19] Goutam Bhat, Martin Danelljan, Luc Van Gool, et al. „Learning discriminative model prediction for tracking.“ 2019. [↰ 2, 61, 62, 64, 66, 104]

[BKC17] Vijay Badrinarayanan, Alex Kendall, and Roberto Cipolla. „Segnet: A deep convolutional encoder-decoder architecture for image segmentation.“ 2017. [↰ 35]

[BKH16] Jimmy Lei Ba, Jamie Ryan Kiros, and Geoffrey E Hinton. „Layer normalization.“ 2016. [↰ 36, 61]

[BKK22] Jongbeom Baek, Gyeongnyeon Kim, and Seungryong Kim. „Semi-Supervised Learning with Mutual Distillation for Monocular Depth Estimation.“ 2022. [↰ 31]

[Bla+23] Philippe Blatter, Menelaos Kanakis, Martin Danelljan, et al. „Efficient Visual Tracking with Exemplar Transformers.“ 2023. [↰ xi, 7, 68, 80]

[Bra+19] Felix JS Bragman, Ryutaro Tanno, Sebastien Ourselin, et al. „Stochastic filter groups for multi-task cnns: Learning specialist and generalist convolution kernels.“ 2019. [↰ 10, 12]

[Bro+21] Andy Brock, Soham De, Samuel L Smith, et al. „High-performance large-scale image recognition without normalization.“ 2021. [↰ 2]

[Bru+20] David Bruggemann, Menelaos Kanakis, Stamatios Georgoulis, et al. „Automated search for resource-efficient branched multi-task networks.“ 2020. [↰ xi, 5, 10, 12, 36, 103]

[Bru+21] David Bruggemann, Menelaos Kanakis, Anton Obukhov, et al. „Exploring relational context for multi-task dense prediction.“ 2021. [↰ xii, 5, 11, 12, 31, 52, 54]

[BTG06] Herbert Bay, Tinne Tuytelaars, and Luc Van Gool. „Surf: Speeded up robust features.“ 2006. [↰ 1, 77–79, 91]

[BV17] Hakan Bilen and Andrea Vedaldi. „Universal representations: The missing link between faces, text, planktons, and cat breeds.“ 2017. [↰ 5, 11, 23]

[Cal+10] Michael Calonder, Vincent Lepetit, Christoph Strecha, et al. „Brief: Binary robust independent elementary features.“ 2010. [↰ 77, 79]

[Can86] John Canny. „A computational approach to edge detection.“ 1986. [↰ 1]

[Cao+17] Zhe Cao, Tomas Simon, Shih-En Wei, et al. „Realtime Multi-Person 2D Pose Estimation using Part Affinity Fields.“ 2017. [↰ 2, 104]

[Cao+19] Z. Cao, G. Hidalgo Martinez, T. Simon, et al. „OpenPose: Realtime Multi-Person 2D Pose Estimation using Part Affinity Fields.“ 2019. [↰ 2, 104]

[Car+20] Nicolas Carion, Francisco Massa, Gabriel Synnaeve, et al. „End-to-end object detection with transformers.“ 2020. [↰ 52, 54, 55, 57]

[Car97] Rich Caruana. „Multitask learning.“ 1997. [↰ 5, 10, 12, 29, 31]

[Car98] Rich Caruana. „A dozen tricks with multitask learning.“ 1998. [↰ 29]

[CBD15] Matthieu Courbariaux, Yoshua Bengio, and Jean-Pierre David. „Binaryconnect: Training deep neural networks with binary weights during propagations.“ 2015. [↰ 4]

[CH19] Jang Hyun Cho and Bharath Hariharan. „On the efficacy of knowledge distillation.“ 2019. [↰ 4]

[Che+14] Xianjie Chen, Roozbeh Mottaghi, Xiaobai Liu, et al. „Detect what you can: Detecting and representing objects using holistic models and body parts.“ 2014. [↰ 19]

[Che+17] Liang-Chieh Chen, George Papandreou, Iasonas Kokkinos, et al. „Deeplab: Semantic image segmentation with deep convolutional nets, atrous convolution, and fully connected crfs.“ 2017. [↰ 2, 9, 19, 37]

[Che+18a] Liang-Chieh Chen, Yukun Zhu, George Papandreou, et al. „Encoder-decoder with atrous separable convolution for semantic image segmentation.“ 2018. [↰ 2, 19, 35, 36]

[Che+18b] Zhao Chen, Vijay Badrinarayanan, Chen-Yu Lee, et al. „Gradnorm: Gradient normalization for adaptive loss balancing in deep multitask networks.“ 2018. [↰ 11, 12, 50]

[Che+19] Po-Yi Chen, Alexander H Liu, Yen-Cheng Liu, et al. „Towards scene understanding: Unsupervised monocular depth estimation with semantic-aware representation.“ 2019. [↰ 29, 42]

[Che+20a] Ting Chen, Simon Kornblith, Mohammad Norouzi, et al. „A simple framework for contrastive learning of visual representations.“ 2020. [↰ 32, 36]

[Che+20b] Xinlei Chen, Haoqi Fan, Ross Girshick, et al. „Improved baselines with momentum contrastive learning.“ 2020. [↰ 30, 35, 36, 38]

[Che+21] Xin Chen, Bin Yan, Jiawen Zhu, et al. „Transformer tracking.“ 2021. [↰ 51, 54, 55, 61, 64]

[Cho+16] Christopher B Choy, JunYoung Gwak, Silvio Savarese, et al. „Universal Correspondence Network.“ 2016. [↰ 79]

[Cho+20] Krzysztof Choromanski, Valerii Likhosherstov, David Dohan, et al. „Masked language modeling for proteins via linearly scalable long-context transformers.“ 2020. [↰ 55]

[Chr+19] Peter Hviid Christiansen, Mikkel Fly Kragh, Yury Brodskiy, et al. „Unsuperpoint: End-to-end unsupervised interest point detector and descriptor.“ 2019. [↰ 81, 85, 86]

[Csu+04] Gabriella Csurka, Christopher Dance, Lixin Fan, et al. „Visual categorization with bags of keypoints.“ 2004. [↰ 1]

[Cus+19] Bart Custers, Alan M Sears, Francien Dechesne, et al. *EU personal data protection in policy and practice*. 2019. [↰ 3]

[CV95] Corinna Cortes and Vladimir Vapnik. „Support-vector networks.“ 1995. [↰ 1]

[Dai+21] Zihang Dai, Hanxiao Liu, Quoc V Le, et al. „Coatnet: Marrying convolution and attention for all data sizes.“ 2021. [↰ 2]

[Dan+] Martin Danelljan, Goutam Bhat, Christoph Mayer, et al. *PyTracking*. [↰ 66]

[Dan+16] Martin Danelljan, Gustav Häger, Fahad Shahbaz Khan, et al. „Discriminative scale space tracking.“ 2016. [↰ 54]

[Dan+17] Martin Danelljan, Goutam Bhat, Fahad Shahbaz Khan, et al. „Eco: Efficient convolution operators for tracking.“ 2017. [↰ 61, 64]

[Dan+19] Martin Danelljan, Goutam Bhat, Fahad Shahbaz Khan, et al. „Atom: Accurate tracking by overlap maximization.“ 2019. [↰ 61, 64, 66]

[Dau+17] Yann N Dauphin, Angela Fan, Michael Auli, et al. „Language modeling with gated convolutional networks.“ 2017. [↰ 17]

[Dav+07] Andrew J Davison, Ian D Reid, Nicholas D Molton, et al. „MonoSLAM: Real-time single camera SLAM.“ 2007. [↰ 92]

[DD17] Koustabh Dolui and Soumya Kanti Datta. „Comparison of edge computing implementations: Fog computing, cloudlet and mobile edge computing.“ 2017. [↰ 3]

[Den+09] Jia Deng, Wei Dong, Richard Socher, et al. „Imagenet: A large-scale hierarchical image database.“ 2009. [↰ 17, 32, 60]

[Den+14] Emily L Denton, Wojciech Zaremba, Joan Bruna, et al. „Exploiting linear structure within convolutional networks for efficient evaluation.“ 2014. [↰ 4, 13, 18]

[Den+19] Jiankang Deng, Jia Guo, Niannan Xue, et al. „Arcface: Additive angular margin loss for deep face recognition.“ 2019. [↰ 82]

[DGE15] Carl Doersch, Abhinav Gupta, and Alexei A Efros. „Unsupervised visual representation learning by context prediction.“ 2015. [↰ 32]

[DGT20] Martin Danelljan, Luc Van Gool, and Radu Timofte. „Probabilistic regression for visual tracking.“ 2020. [↰ 61, 64]

[DGZ21] Weijian Deng, Stephen Gould, and Liang Zheng. „What does rotation prediction tell us about classifier accuracy under varying testing environments?“ 2021. [↰ 32]

[DHS16] Jifeng Dai, Kaiming He, and Jian Sun. „Instance-aware semantic segmentation via multi-task network cascades.“ 2016. [↰ 29]

[DMR18] Daniel DeTone, Tomasz Malisiewicz, and Andrew Rabinovich. „Superpoint: Self-supervised interest point detection and description.“ 2018. [↰ 75, 76, 79, 81, 85, 86, 91, 93, 98]

[Dos+21] Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, et al. „An image is worth 16x16 words: Transformers for image recognition at scale.“ 2021. [↰ 2, 55]

[DR19] Kshitij Dwivedi and Gemma Roig. „Representation similarity analysis for efficient task taxonomy & transfer learning.“ 2019. [↰ 15]

[DT05] Navneet Dalal and Bill Triggs. „Histograms of oriented gradients for human detection.“ 2005. [↰ 1]

[Dus+19] Mihai Dusmanu, Ignacio Rocco, Tomas Pajdla, et al. „D2-net: A trainable cnn for joint description and detection of local features.“ 2019. [↰ 75, 79, 81]

[DZ17] Carl Doersch and Andrew Zisserman. „Multi-task self-supervised visual learning.“ 2017. [↰ 10, 12]

[EH10] Ian Endres and Derek Hoiem. „Category independent object proposals.“ 2010. [↰ 48]

[End+12] Felix Endres, Jürgen Hess, Nikolas Engelhard, et al. „An evaluation of the RGB-D SLAM system.“ 2012. [↰ 75, 92]

[EPF14] David Eigen, Christian Puhrsch, and Rob Fergus. „Depth map prediction from a single image using a multi-scale deep network.“ 2014. [↰ 9]

[Eve+10] Mark Everingham, Luc Van Gool, Christopher KI Williams, et al. „The pascal visual object classes (voc) challenge.“ 2010. [↰ 18, 43]

[Fan+19] Heng Fan, Liting Lin, Fan Yang, et al. „Lasot: A high-quality benchmark for large-scale single object tracking.“ 2019. [↰ 51, 53, 61, 66, 68, 69]

[Fat+18] Mohammed E Fathy, Quoc-Huy Tran, M Zeeshan Zia, et al. „Hierarchical metric learning and matching for 2d and 3d geometric correspondences.“ 2018. [↰ 79]

[Fre99] Robert M French. „Catastrophic forgetting in connectionist networks.“ 1999. [↰ 13, 103]

[Fu+18] Huan Fu, Mingming Gong, Chaohui Wang, et al. „Deep ordinal regression network for monocular depth estimation.“ 2018. [↰ 2, 32]

[Gar+16] Timur Garipov, Dmitry Podoprikhin, Alexander Novikov, et al. „Ultimate tensorization: compressing convolutional and fc layers alike.“ 2016. [↰ 4]

[Gei+18] Robert Geirhos, Patricia Rubisch, Claudio Michaelis, et al. „ImageNet-trained CNNs are biased towards texture; increasing shape bias improves accuracy and robustness.“ 2018. [↰ 47]

[Geo+21] Mariana-Iuliana Georgescu, Antonio Barbalau, Radu Tudor Ionescu, et al. „Anomaly detection in video via self-supervised and multi-task learning.“ 2021. [↰ 31]

[GHC21] Stephen Gould, Richard Hartley, and Dylan John Campbell. „Deep declarative networks.“ 2021. [↰ 83]

[Ghi+21] Golnaz Ghiasi, Barret Zoph, Ekin D Cubuk, et al. „Multi-task self-training for learning general representations.“ 2021. [↰ 32]

[Gho+18] Amir Gholami, Kiseok Kwon, Bichen Wu, et al. „Squeezenext: Hardware-aware neural network design.“ 2018. [↰ 4]

[Gid+19] Spyros Gidaris, Andrei Bursuc, Nikos Komodakis, et al. „Boosting few-shot visual learning with self-supervision.“ 2019. [↰ 33, 34, 43]

[Gir+14] Ross Girshick, Jeff Donahue, Trevor Darrell, et al. „Rich feature hierarchies for accurate object detection and semantic segmentation.“ 2014. [↰ 2, 9, 104]

[Gir15] Ross Girshick. „Fast r-cnn.“ 2015. [↰ 2, 104]

[GLU12] Andreas Geiger, Philip Lenz, and Raquel Urtasun. „Are we ready for autonomous driving? the kitti vision benchmark suite.“ 2012. [↰ 95]

[God+19] Clément Godard, Oisin Mac Aodha, Michael Firman, et al. „Digging into self-supervised monocular depth estimation.“ 2019. [↰ 2]

[Gor+18] Ariel Gordon, Elad Eban, Ofir Nachum, et al. „Morphnet: Fast & simple resource-constrained structure learning of deep networks.“ 2018. [↰ 4]

[Goy+21] Priya Goyal, Quentin Duval, Jeremy Reizenstein, et al. *VISSL*. 2021. [↰ 37]

[GSK18] Spyros Gidaris, Praveer Singh, and Nikos Komodakis. „Unsupervised representation learning by predicting image rotations.“ 2018. [↰ 30, 32, 34]

[GT12] Dorian Gálvez-López and Juan D Tardos. „Bags of binary words for fast place recognition in image sequences.“ 2012. [↰ 75, 92]

[GT20] Lukas Geiger and Plumerai Team. „Larq: An Open-Source Library for Training Binarized Neural Networks.“ Jan. 2020. [↰ 85, 88]

[Gui+20a] Vitor Guizilini, Rui Hou, Jie Li, et al. „Semantically-guided representation learning for self-supervised monocular depth.“ 2020. [↰ 42]

[Gui+20b] Vitor Guizilini, Jie Li, Rares Ambrus, et al. „Robust semi-supervised monocular depth estimation with reprojected distances.“ 2020. [↰ 31]

[Guo+17] Qing Guo, Wei Feng, Ce Zhou, et al. „Learning dynamic siamese network for visual object tracking.“ 2017. [↰ 54]

[Guo+18] Michelle Guo, Albert Haque, De-An Huang, et al. „Dynamic task prioritization for multitask learning.“ 2018. [↰ 11, 12]

[GZX19] Junyu Gao, Tianzhu Zhang, and Changsheng Xu. „Graph convolutional tracking.“ 2019. [↰ 54]

[Han+15] Song Han, Jeff Pool, John Tran, et al. „Learning both weights and connections for efficient neural network.“ 2015. [↰ 4]

[Har+11] Bharath Hariharan, Pablo Arbeláez, Lubomir Bourdev, et al. „Semantic contours from inverse detectors.“ 2011. [↰ 38, 43]

[Har93] Chris Harris. „Geometry from visual motion.“ 1993. [↰ 1]

[HCL06] Raia Hadsell, Sumit Chopra, and Yann LeCun. „Dimensionality reduction by learning an invariant mapping.“ 2006. [↰ 82]

[HD19] Dan Hendrycks and Thomas Dietterich. „Benchmarking neural network robustness to common corruptions and perturbations.“ 2019. [↰ 33]

[HDF12] Jared Heinly, Enrique Dunn, and Jan-Michael Frahm. „Comparative evaluation of binary features.“ 2012. [↰ 77, 79]

[He+15] Kaiming He, Xiangyu Zhang, Shaoqing Ren, et al. „Delving deep into rectifiers: Surpassing human-level performance on imagenet classification.“ 2015. [↰ 61]

[He+16] Kaiming He, Xiangyu Zhang, Shaoqing Ren, et al. „Deep residual learning for image recognition.“ 2016. [↰ 2, 4, 9, 15, 19, 36, 57, 71, 72]

[He+17] Kaiming He, Georgia Gkioxari, Piotr Dollár, et al. „Mask r-cnn.“ 2017. [↰ 105]

[He+20] Kaiming He, Haoqi Fan, Yuxin Wu, et al. „Momentum contrast for unsupervised visual representation learning.“ 2020. [↰ 30, 32, 35, 36, 38, 103]

[Hen+14] João F Henriques, Rui Caseiro, Pedro Martins, et al. „High-speed tracking with kernelized correlation filters.“ 2014. [↰ 54, 66]

[Hen+19] Dan Hendrycks, Mantas Mazeika, Saurav Kadavath, et al. „Using self-supervised learning can improve model robustness and uncertainty.“ 2019. [↰ 33, 35, 44]

[Hen20] Olivier Henaff. „Data-efficient image recognition with contrastive predictive coding.“ 2020. [↰ 36, 103]

[HGD19] Kaiming He, Ross Girshick, and Piotr Dollár. „Rethinking imagenet pre-training.“ 2019. [↰ 32]

[HMD16] Song Han, Huizi Mao, and William J Dally. „Deep compression: Compressing deep neural networks with pruning, trained quantization and huffman coding.“ 2016. [↰ 4]

[How+17] Andrew G Howard, Menglong Zhu, Bo Chen, et al. „Mobilenets: Efficient convolutional neural networks for mobile vision applications.“ 2017. [↰ 4, 60, 103, 105]

[How+19] Andrew Howard, Mark Sandler, Grace Chu, et al. „Searching for mobilenetv3.“ 2019. [↰ 4, 71, 72, 80, 86]

[Hoy+21] Lukas Hoyer, Dengxin Dai, Yuhua Chen, et al. „Three ways to improve semantic segmentation with self-supervised depth estimation.“ 2021. [↰ 5, 31]

[HS+88] Chris Harris, Mike Stephens, et al. „A combined corner and edge detector.“ 1988. [↰ 1]

[HS15] Kaiming He and Jian Sun. „Convolutional neural networks at constrained time cost.“ 2015. [↰ 80]

[HSS18] Jie Hu, Li Shen, and Gang Sun. „Squeeze-and-excitation networks.“ 2018. [↰ 24]

[Hua+16] Gao Huang, Yu Sun, Zhuang Liu, et al. „Deep networks with stochastic depth.“ 2016. [↰ 104]

[Hua+18] Gao Huang, Shichen Liu, Laurens Van der Maaten, et al. „Condensenet: An efficient densenet using learned group convolutions.“ 2018. [↰ 4]

[HVD15] Geoffrey Hinton, Oriol Vinyals, and Jeff Dean. „Distilling the Knowledge in a Neural Network.“ 2015. [↰ 4]

[HZ03] Richard Hartley and Andrew Zisserman. *Multiple view geometry in computer vision*. 2003. [↰ 95]

[HZH19] Lianghua Huang, Xin Zhao, and Kaiqi Huang. „Got-10k: A large high-diversity benchmark for generic object tracking in the wild.“ 2019. [↰ 61]

[Ign+18] Andrey Ignatov, Radu Timofte, William Chou, et al. „Ai benchmark: Running deep neural networks on android smartphones.“ 2018. [↰ 4]

[IS15] Sergey Ioffe and Christian Szegedy. „Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift.“ 2015. [↰ 18, 21, 36, 60]

[Jac18] Benoit Jacob et al. „Quantization and Training of Neural Networks for Efficient Integer-Arithmetic-Only Inference.“ 2018. [↰ 4, 80]

[Jia+18] Jianbo Jiao, Ying Cao, Yibing Song, et al. „Look deeper into depth: Monocular depth estimation with semantic booster and attention-driven loss.“ 2018. [↰ 42]

[JVZ14] Max Jaderberg, Andrea Vedaldi, and Andrew Zisserman. „Speeding up Convolutional Neural Networks with Low Rank Expansions.“ 2014. [↰ 4, 13, 103]

[Kan+20] Menelaos Kanakis, David Bruggemann, Suman Saha, et al. „Reparameterizing convolutions for incremental multi-task learning without task interference.“ 2020. [↰ xi, 6, 31]

[Kan+22] Menelaos Kanakis, Simon Maurer, Matteo Spallanzani, et al. „ZippyPoint: Fast Interest Point Detection, Description, and Matching through Mixed Precision Discretization.“ 2022. [↰ xi, 7]

[Kan+23] Menelaos Kanakis, Thomas E Huang, David Bruggemann, et al. „Composite Learning for Robust and Effective Dense Predictions.“ 2023. [↰ xi, 6]

[Kat+20] Angelos Katharopoulos, Apoorv Vyas, Nikolaos Pappas, et al. „Transformers are rnns: Fast autoregressive transformers with linear attention.“ 2020. [↰ 55, 72, 73]

[KB15] Diederik P Kingma and Jimmy Ba. „Adam: A method for stochastic optimization.“ 2015. [↰ 85]

[KGC18] Alex Kendall, Yarin Gal, and Roberto Cipolla. „Multi-task learning using uncertainty to weigh losses for scene geometry and semantics.“ 2018. [↰ 11, 12, 20]

[Kho+20] Prannay Khosla, Piotr Teterwak, Chen Wang, et al. „Supervised contrastive learning.“ 2020. [↰ 32, 38]

[Kia+17] Hamed Kiani Galoogahi, Ashton Fagg, Chen Huang, et al. „Need for speed: A benchmark for higher frame rate object tracking.“ 2017. [↰ 51, 53, 61, 62, 68, 69]

[Kim+16] Yong-Deok Kim, Eunhyeok Park, Sungjoo Yoo, et al. „Compression of deep convolutional neural networks for fast and low power mobile applications.“ 2016. [↰ 4, 13]

[Kir+17] James Kirkpatrick, Razvan Pascanu, Neil Rabinowitz, et al. „Overcoming catastrophic forgetting in neural networks.“ 2017. [↰ 13, 103]

[KKL20] Nikita Kitaev, Łukasz Kaiser, and Anselm Levskaya. „Reformer: The efficient transformer.“ 2020. [↰ 55]

[Kok16] Iasonas Kokkinos. „Pushing the boundaries of boundary detection using deep learning.“ 2016. [↰ 20]

[Kok17] Iasonas Kokkinos. „Ubernet: Training a universal convolutional neural network for low-, mid-, and high-level vision using diverse datasets and limited memory.“ 2017. [↰ 5, 10–12, 29]

[Kri+20] Matej Kristan, Aleš Leonardis, Jiří Matas, et al. „The eighth visual object tracking VOT2020 challenge results.“ 2020. [↰ 51, 53, 61, 63]

[KS15] Minje Kim and Paris Smaragdis. „Bitwise neural networks.“ 2015. [↰ 4]

[KSH12] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. „ImageNet Classification with Deep Convolutional Neural Networks.“ 2012. [↰ 2, 9, 32]

[KSS11] Laurent Kneip, Davide Scaramuzza, and Roland Siegwart. „A novel parametrization of the perspective-three-point problem for a direct computation of absolute camera position and orientation.“ 2011. [↰ 92]

[Lai+15] Hanjiang Lai, Yan Pan, Ye Liu, et al. „Simultaneous feature learning and hash coding with deep neural networks.“ 2015. [↰ 78, 79, 90]

[Lai+16] Iro Laina, Christian Rupprecht, Vasileios Belagiannis, et al. „Deeper depth prediction with fully convolutional residual networks.“ 2016. [↰ 9]

[LBH15] Yann LeCun, Yoshua Bengio, and Geoffrey Hinton. „Deep learning.“ 2015. [↰ 1]

[LCS11] Stefan Leutenegger, Margarita Chli, and Roland Y Siegwart. „BRISK: Binary robust invariant scalable keypoints.“ 2011. [↰ 75, 77, 79, 90, 91, 93, 98]

[LDJ19] Shikun Liu, Andrew Davison, and Edward Johns. „Self-supervised generalisation with meta auxiliary learning.“ 2019. [↰ 31, 103]

[Leb+15] Vadim Lebedev, Yaroslav Ganin, Maksim Rakhuba, et al. „Speeding-up Convolutional Neural Networks Using Finetuned CP-Decomposition.“ 2015. [↰ 4, 13]

[LeC+89] Yann LeCun, Bernhard Boser, John Denker, et al. „Handwritten digit recognition with a back-propagation network.“ 1989. [↰ 2]

[Lee+17] Sang-Woo Lee, Jin-Hwa Kim, Jaehyun Jun, et al. „Overcoming catastrophic forgetting by incremental moment matching.“ 2017. [↰ 13, 103]

[Leu+15] Stefan Leutenegger, Simon Lynen, Michael Bosse, et al. „Keyframe-based visual–inertial odometry using nonlinear optimization.“ 2015. [↰ 75, 79, 92]

[LH17] Zhizhong Li and Derek Hoiem. „Learning without forgetting.“ 2017. [↰ 13, 103]

[Li+17] Hao Li, Asim Kadav, Igor Durdanovic, et al. „Pruning filters for efficient convnets.“ 2017. [↰ 4]

[Li+18] Bo Li, Junjie Yan, Wei Wu, et al. „High performance visual tracking with siamese region proposal network.“ 2018. [↰ 51, 53]

[Li+19a] Bo Li, Wei Wu, Qiang Wang, et al. „Siamrpn++: Evolution of siamese visual tracking with very deep networks.“ 2019. [↰ 61, 64, 65]

[Li+19b] Yawei Li, Shuhang Gu, Luc Van Gool, et al. „Learning Filter Basis for Convolutional Neural Network Compression.“ 2019. [↰ 13]

[Li+22] Zhaowen Li, Yousong Zhu, Fan Yang, et al. „UniVIP: A Unified Framework for Self-Supervised Visual Pre-training.“ 2022. [↰ 32]

[Lin+14] Tsung-Yi Lin, Michael Maire, Serge Belongie, et al. „Microsoft coco: Common objects in context.“ 2014. [↰ 45, 61, 85]

[Lin+16] Kevin Lin, Jiwen Lu, Chu-Song Chen, et al. „Learning compact binary descriptors with unsupervised deep neural networks.“ 2016. [↰ 79]

[Lin+17] Tsung-Yi Lin, Piotr Dollár, Ross Girshick, et al. „Feature pyramid networks for object detection.“ 2017. [↰ 20]

[Liu+12] Wei Liu, Jun Wang, Rongrong Ji, et al. „Supervised hashing with kernels.“ 2012. [↰ 90]

[Liu+16] Wei Liu, Dragomir Anguelov, Dumitru Erhan, et al. „Ssd: Single shot multibox detector.“ 2016. [↰ 9]

[Liu+18a] Chenxi Liu, Barret Zoph, Maxim Neumann, et al. „Progressive neural architecture search.“ 2018. [↰ 2, 4]

[Liu+18b] Zechun Liu, Baoyuan Wu, Wenhan Luo, et al. „Bi-real net: Enhancing the performance of 1-bit cnns with improved representational capability and advanced training algorithm.“ 2018. [↰ 4, 77, 80, 86, 88]

[Liu+21] Ze Liu, Yutong Lin, Yue Cao, et al. „Swin transformer: Hierarchical vision transformer using shifted windows.“ 2021. [↰ 55, 72, 73]

[LJD19] Shikun Liu, Edward Johns, and Andrew J Davison. „End-to-end multi-task learning with attention.“ 2019. [↰ 10, 12]

[Low04] David G Lowe. „Distinctive image features from scale-invariant keypoints.“ 2004. [↰ 1, 77–79, 91–93, 95, 98]

[LS20] Gian Paolo Leonardi and Matteo Spallanzani. „Analytical aspects of non-differentiable neural networks.“ 2020. [↰ 4, 78, 80]

[LSD15] Jonathan Long, Evan Shelhamer, and Trevor Darrell. „Fully convolutional networks for semantic segmentation.“ 2015. [↰ 9, 32]

[LSY18] Hanxiao Liu, Karen Simonyan, and Yiming Yang. „DARTS: Differentiable Architecture Search.“ 2018. [↰ 4]

[Lu+17] Yongxi Lu, Abhishek Kumar, Shuangfei Zhai, et al. „Fully-adaptive feature sharing in multi-task networks with applications in person attribute classification.“ 2017. [↰ 10, 12, 14]

[LZP17] Xiaofan Lin, Cong Zhao, and Wei Pan. „Towards accurate binary convolutional neural network.“ 2017. [↰ 4]

[Ma+18] Ningning Ma, Xiangyu Zhang, Hai-Tao Zheng, et al. „Shufflenet v2: Practical guidelines for efficient cnn architecture design.“ 2018. [↰ 4]

[MA16] Andre Martins and Ramon Astudillo. „From softmax to sparsemax: A sparse model of attention and multi-label classification.“ 2016. [↰ 83]

[Man+17] Kevis-Kokitsi Maninis, Jordi Pont-Tuset, Pablo Arbeláez, et al. „Convolutional oriented boundaries: From image segmentation to high-level tasks.“ 2017. [↰ 20]

[Mar+15] Martín Abadi, Ashish Agarwal, Paul Barham, et al. *TensorFlow: Large-Scale Machine Learning on Heterogeneous Systems*. 2015. [↰ 4, 85]

[Mar76] David Marr. „Analyzing natural images: A computational theory of texture vision.“ 1976. [↰ 1]

[Mar82] David Marr. „Vision: A Computational Investigation into the Human Representation and Processing of Visual Information.“ 1982. [↰ 1]

[May+22] Christoph Mayer, Martin Danelljan, Goutam Bhat, et al. „Transforming model prediction for tracking.“ 2022. [↰ 54]

[MBL20] Kevin Musgrave, Serge Belongie, and Ser-Nam Lim. „A metric learning reality check.“ 2020. [↰ 78, 79, 82]

[MDL18] Arun Mallya, Dillon Davis, and Svetlana Lazebnik. „Piggyback: Adapting a single network to multiple tasks by learning to mask weights.“ 2018. [↰ 13]

[MFM04] David R Martin, Charless C Fowlkes, and Jitendra Malik. „Learning to detect natural image boundaries using local brightness, color, and texture cues.“ 2004. [↰ 19, 20, 47]

[MHN+13] Andrew L Maas, Awni Y Hannun, Andrew Y Ng, et al. „Rectifier nonlinearities improve neural network acoustic models.“ 2013. [↰ 81]

[Mis+16] Ishan Misra, Abhinav Shrivastava, Abhinav Gupta, et al. „Cross-stitch networks for multi-task learning.“ 2016. [↰ 5, 10, 12, 14]

[MM18] Asit Mishra and Debbie Marr. „Apprentice: Using Knowledge Distillation Techniques To Improve Low-Precision Network Accuracy.“ 2018. [↰ 4]

[MMT15] Raul Mur-Artal, Jose Maria Martinez Montiel, and Juan D Tardos. „ORB-SLAM: a versatile and accurate monocular SLAM system.“ 2015. [↰ 75, 79, 90, 92]

[Mor81] Hans P Moravec. „Rover Visual Obstacle Avoidance.“ 1981. [↰ 1]

[Mot+14] Roozbeh Mottaghi, Xianjie Chen, Xiaobai Liu, et al. „The role of context for object detection and semantic segmentation in the wild.“ 2014. [↰ 9, 15, 18, 20, 101]

[MRK19] Kevis-Kokitsi Maninis, Ilija Radosavovic, and Iasonas Kokkinos. „Attentive single-tasking of multiple tasks.“ 2019. [↰ 5, 11, 15, 16, 19–21, 23–25, 27, 28, 31, 36, 49]

[MSG16] Matthias Mueller, Neil Smith, and Bernard Ghanem. „A benchmark and simulator for uav tracking.“ 2016. [↰ 53, 61, 63, 68]

[MT17] Raul Mur-Artal and Juan D Tardós. „Orb-slam2: An open-source slam system for monocular, stereo, and rgb-d cameras.“ 2017. [↰ 75, 79, 92]

[Mul+18] Matthias Muller, Adel Bibi, Silvio Giancola, et al. „Trackingnet: A large-scale dataset and benchmark for object tracking in the wild.“ 2018. [↰ 51, 53, 61, 63]

[Nag+19] Markus Nagel, Mart van Baalen, Tijmen Blankevoort, et al. „Data-free quantization through weight equalization and bias correction.“ 2019. [↰ 4, 77, 80]

[ND20] Alejandro Newell and Jia Deng. „How useful is self-supervised pretraining for visual tasks?“ 2020. [↰ 32]

[Nev+17] Davy Neven, Bert De Brabandere, Stamatios Georgoulis, et al. „Fast scene understanding for autonomous driving.“ 2017. [↰ 10, 12]

[NFS12] Mohammad Norouzi, David J Fleet, and Russ R Salakhutdinov. „Hamming distance metric learning.“ 2012. [↰ 79]

[Nis04] David Nistér. „An efficient solution to the five-point relative pose problem.“ 2004. [↰ 92, 95]

[Nov+15] Alexander Novikov, Dmitrii Podoprikhin, Anton Osokin, et al. „Tensorizing neural networks.“ 2015. [↰ 4]

[NYD16] Alejandro Newell, Kaiyu Yang, and Jia Deng. „Stacked hourglass networks for human pose estimation.“ 2016. [↰ 2, 104]

[Obu+20] Anton Obukhov, Maxim Rakhuba, Stamatios Georgoulis, et al. „T-basis: a compact representation for neural networks.“ 2020. [↰ xi, 13, 80]

[Obu+21] Anton Obukhov, Maxim Rakhuba, Alexander Liniger, et al. „Spectral tensor train parameterization of deep learning layers.“ 2021. [↰ 80]

[OLV18] Aaron van den Oord, Yazhe Li, and Oriol Vinyals. „Representation learning with contrastive predictive coding.“ 2018. [↰ 35, 38]

[Ono+18] Yuki Ono, Eduard Trulls, Pascal Fua, et al. „LF-Net: Learning local features from images.“ 2018. [↰ 79]

[Ose11] Ivan Oseledets. „Tensor-train decomposition.“ 2011. [↰ 4, 13]

[Pan+21] Jiangmiao Pang, Linlu Qiu, Xia Li, et al. „Quasi-dense similarity learning for multiple object tracking.“ 2021. [↰ 32]

[Par+18] Niki Parmar, Ashish Vaswani, Jakob Uszkoreit, et al. „Image transformer.“ 2018. [↰ 55]

[Pas+19] Adam Paszke, Sam Gross, Francisco Massa, et al. „Pytorch: An imperative style, high-performance deep learning library.“ 2019. [↰ 4, 17]

[Pen+18] Bo Peng, Wenming Tan, Zheyang Li, et al. „Extreme network compression via filter group approximation.“ 2018. [↰ 13]

[PM15] Jordi Pont-Tuset and Ferran Marques. „Supervised evaluation of image segmentation and object proposal techniques.“ 2015. [↰ 20]

[PWC] PWC. *Papers With Code: Image Classification on ImageNet.* [↰ 2]

[Rad+20] Ilija Radosavovic, Raj Prateek Kosaraju, Ross Girshick, et al. „Designing network design spaces.“ 2020. [↰ 4]

[Ram+19] Prajit Ramachandran, Niki Parmar, Ashish Vaswani, et al. „Stand-alone self-attention in vision models.“ 2019. [↰ 55, 72, 73]

[Ran+19] Anurag Ranjan, Varun Jampani, Lukas Balles, et al. „Competitive collaboration: Joint unsupervised learning of depth, camera motion, optical flow and motion segmentation.“ 2019. [↰ 5]

[Ras+16] Mohammad Rastegari, Vicente Ordonez, Joseph Redmon, et al. „Xnor-net: Imagenet classification using binary convolutional neural networks.“ 2016. [↰ 4, 77, 80, 86–88]

[RBK21] René Ranftl, Alexey Bochkovskiy, and Vladlen Koltun. „Vision transformers for dense prediction.“ 2021. [↰ 95]

[RBV17] Sylvestre-Alvise Rebuffi, Hakan Bilen, and Andrea Vedaldi. „Learning multiple visual domains with residual adapters.“ 2017. [↰ 5, 11, 13, 24, 25, 27]

[RBV18] Sylvestre-Alvise Rebuffi, Hakan Bilen, and Andrea Vedaldi. „Efficient parametrization of multi-domain deep neural networks.“ 2018. [↰ 5, 11, 13, 24–27]

[Rea+19] Esteban Real, Alok Aggarwal, Yanping Huang, et al. „Regularized evolution for image classifier architecture search.“ 2019. [↰ 2]

[Reb+17] Sylvestre-Alvise Rebuffi, Alexander Kolesnikov, Georg Sperl, et al. „icarl: Incremental classifier and representation learning.“ 2017. [↰ 13, 103]

[Red+16] Joseph Redmon, Santosh Divvala, Ross Girshick, et al. „You only look once: Unified, real-time object detection.“ 2016. [↰ 2, 9, 104]

[Ren+15] Shaoqing Ren, Kaiming He, Ross Girshick, et al. „Faster r-cnn: Towards real-time object detection with region proposal networks.“ 2015. [↰ 2, 104]

[Rev+19] Jerome Revaud, Philippe Weinzaepfel, César De Souza, et al. „R2D2: repeatable and reliable detector and descriptor.“ 2019. [↰ 79, 81, 92]

[RFB15] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. „U-net: Convolutional networks for biomedical image segmentation.“ 2015. [↰ 35]

[Rob63] Lawrence G Roberts. „Machine perception of three-dimensional solids.“ 1963. [↰ 1]

[Rom+15] Adriana Romero, Nicolas Ballas, Samira Ebrahimi Kahou, et al. „Fitnets: Hints for thin deep nets.“ 2015. [↰ 4]

[RPC17] Rajeev Ranjan, Vishal M Patel, and Rama Chellappa. „Hyperface: A deep multi-task learning framework for face detection, landmark localization, pose estimation, and gender recognition.“ 2017. [↰ 5, 31]

[RT18] Amir Rosenfeld and John K Tsotsos. „Incremental learning through deep adaptation.“ 2018. [↰ 13]

[Rub+11] Ethan Rublee, Vincent Rabaud, Kurt Konolige, et al. „ORB: An efficient alternative to SIFT or SURF.“ 2011. [↰ 75, 77, 79, 91, 93, 98]

[Rud16] Sebastian Ruder. „An overview of gradient descent optimization algorithms.“ 2016. [↰ 61]

[Rud17] Sebastian Ruder. „An overview of multi-task learning in deep neural networks.“ 2017. [↰ 10, 12]

[Rus+15] Olga Russakovsky, Jia Deng, Hao Su, et al. „Imagenet large scale visual recognition challenge.“ 2015. [↰ 2]

[Ryo+21] Michael S Ryoo, AJ Piergiovanni, Anurag Arnab, et al. „TokenLearner: What Can 8 Learned Tokens Do for Images and Videos?“ 2021. [↰ 55]

[Sah+20] Suman Saha, Wenhao Xu, Menelaos Kanakis, et al. „Domain agnostic feature learning for image and video based face anti-spoofing.“ 2020. [↰ xi]

[Sah+21] Suman Saha, Anton Obukhov, Danda Pani Paudel, et al. „Learning to relate depth and semantics for unsupervised domain adaptation.“ 2021. [↰ xii, 31]

[San+18] Mark Sandler, Andrew Howard, Menglong Zhu, et al. „Mobilenetv2: Inverted residuals and linear bottlenecks.“ 2018. [↰ 4, 103, 105]

[Sar+19] Paul-Edouard Sarlin, Cesar Cadena, Roland Siegwart, et al. „From coarse to fine: Robust hierarchical localization at large scale.“ 2019. [↰ 92–95]

[Sar+20] Paul-Edouard Sarlin, Daniel DeTone, Tomasz Malisiewicz, et al. „Superglue: Learning feature matching with graph neural networks.“ 2020. [↰ 76, 92]

[Sar+22] Paul-Edouard Sarlin, Mihai Dusmanu, Johannes L Schönberger, et al. „Lamar: Benchmarking localization and mapping for augmented reality.“ 2022. [↰ 3]

[Sat+12] Torsten Sattler, Tobias Weyand, Bastian Leibe, et al. „Image Retrieval for Image-Based Localization Revisited.“ 2012. [↰ 76, 92]

[Sat+18] Torsten Sattler, Will Maddern, Carl Toft, et al. „Benchmarking 6dof outdoor visual localization in changing conditions.“ 2018. [↰ 76, 77, 79, 92]

[SF16] Johannes L Schonberger and Jan-Michael Frahm. „Structure-from-motion revisited.“ 2016. [↰ 79]

[She+15] Fumin Shen, Chunhua Shen, Wei Liu, et al. „Supervised discrete hashing.“ 2015. [↰ 79]

[She+18] Fumin Shen, Yan Xu, Li Liu, et al. „Unsupervised deep hashing with similarity-adaptive and discrete optimization.“ 2018. [↰ 78, 79]

[Sil+12] Nathan Silberman, Derek Hoiem, Pushmeet Kohli, et al. „Indoor segmentation and support inference from rgbd images.“ 2012. [↰ 9, 19, 20, 38, 39, 48, 49, 101]

[Sin+18] Ayan Sinha, Zhao Chen, Vijay Badrinarayanan, et al. „Gradient adversarial training of neural networks.“ 2018. [↰ 11, 12]

[SK16] Tim Salimans and Durk P Kingma. „Weight normalization: A simple reparameterization to accelerate training of deep neural networks.“ 2016. [↰ 17]

[SK18] Ozan Sener and Vladlen Koltun. „Multi-task learning as multi-objective optimization.“ 2018. [↰ 11, 12, 20]

[SM97] Cordelia Schmid and Roger Mohr. „Local grayvalue invariants for image retrieval.“ 1997. [↰ 1]

[Sri+14] Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, et al. „Dropout: a simple way to prevent neural networks from overfitting.“ 2014. [↰ 61, 104]

[SS05] Nathan Srebro and Adi Shraibman. „Rank, trace-norm and max-norm.“ 2005. [↰ 17]

[SS20] Zhiqiang Shen and Marios Savvides. „Meal v2: Boosting vanilla resnet-50 to 80%+ top-1 accuracy on imagenet without tricks.“ 2020. [↰ 4]

[Sta+20] Trevor Standley, Amir Zamir, Dawn Chen, et al. „Which tasks should be learned together in multi-task learning?“ 2020. [↰ 5, 30, 31, 103, 104]

[Su+18] Jiahao Su, Jingling Li, Bobby Bhattacharjee, et al. „Tensorial neural networks: Generalization of neural networks and application to model compression.“ 2018. [↰ 4]

[Sun+19] Ke Sun, Bin Xiao, Dong Liu, et al. „Deep High-Resolution Representation Learning for Human Pose Estimation.“ 2019. [↰ 2, 104]

[Sun+20] Peize Sun, Yi Jiang, Rufeng Zhang, et al. „Transtrack: Multiple-object tracking with transformer.“ 2020. [↰ 52]

[Sun+21] Guolei Sun, Thomas Probst, Danda Pani Paudel, et al. „Task Switching Network for Multi-task Learning.“ 2021. [↰ xii]

[SWP05] Thomas Serre, Lior Wolf, and Tomaso Poggio. „Object recognition with features inspired by visual cortex.“ 2005. [↰ 1]

[SZ15] Karen Simonyan and Andrew Zisserman. „Very deep convolutional networks for large-scale image recognition.“ 2015. [↰ 2, 9, 81]

[Sze+15] Christian Szegedy, Wei Liu, Yangqing Jia, et al. „Going deeper with convolutions.“ 2015. [↰ 2]

[Sze+16] Christian Szegedy, Vincent Vanhoucke, Sergey Ioffe, et al. „Rethinking the inception architecture for computer vision.“ 2016. [↰ 2]

[Tan+19] Jiexiong Tang, Ludvig Ericson, John Folkesson, et al. „GCNv2: Efficient correspondence prediction for real-time SLAM.“ 2019. [↰ 79, 90]

[Tan+20] Jiexiong Tang, Hanme Kim, Vitor Guizilini, et al. „Neural Outlier Rejection for Self-Supervised Keypoint Learning.“ 2020. [↰ 79, 81, 85, 86, 89, 91, 94, 105]

[Tan+22] Yehui Tang, Kai Han, Yunhe Wang, et al. „Patch slimming for efficient vision transformers.“ 2022. [↰ 55]

[Tay+20] Yi Tay, Mostafa Dehghani, Dara Bahri, et al. „Efficient transformers: A survey.“ 2020. [↰ 55]

[Tia+20] Yonglong Tian, Chen Sun, Ben Poole, et al. „What makes for good views for contrastive learning.“ 2020. [↰ 38]

[TL19] Mingxing Tan and Quoc Le. „Efficientnet: Rethinking model scaling for convolutional neural networks.“ 2019. [↰ 4, 103, 105]

[TL21] Mingxing Tan and Quoc Le. „Efficientnetv2: Smaller models and faster training.“ 2021. [↰ 4]

[TLF09] Engin Tola, Vincent Lepetit, and Pascal Fua. „Daisy: An efficient dense descriptor applied to wide-baseline stereo.“ 2009. [↰ 79]

[Tol+21] Ilya O Tolstikhin, Neil Houlsby, Alexander Kolesnikov, et al. „Mlp-mixer: An all-mlp architecture for vision.“ 2021. [↰ 2]

[Tou+19] Hugo Touvron, Andrea Vedaldi, Matthijs Douze, et al. „Fixing the train-test resolution discrepancy.“ 2019. [↰ 2]

[TV17] Antti Tarvainen and Harri Valpola. „Mean teachers are better role models: Weight-averaged consistency targets improve semi-supervised deep learning results.“ 2017. [↰ 4]

[Ull+20] Inam Ullah, Muwei Jian, Sumaira Hussain, et al. „A brief survey of visual saliency detection.“ 2020. [↰ 105]

[UVL16] Dmitry Ulyanov, Andrea Vedaldi, and Victor Lempitsky. „Instance normalization: The missing ingredient for fast stylization.“ 2016. [↰ 36]

[Van+20] Simon Vandenhende, Stamatios Georgoulis, Bert De Brabandere, et al. „Branched multi-task networks: Deciding what layers to share.“ 2020. [↰ 10, 12]

[Vas+17] Ashish Vaswani, Noam Shazeer, Niki Parmar, et al. „Attention is all you need.“ 2017. [↰ 52, 54–56, 70, 72, 73]

[VGG20] Simon Vandenhende, Stamatios Georgoulis, and Luc Van Gool. „Mti-net: Multi-scale task interaction networks for multi-task learning.“ 2020. [↰ 11, 12, 20, 21]

[VH08] Laurens Van der Maaten and Geoffrey Hinton. „Visualizing data using t-SNE.“ 2008. [↰ 41]

[VKF20] Apoorv Vyas, Angelos Katharopoulos, and François Fleuret. „Fast transformers with clustered attention.“ 2020. [↰ 55, 72, 73]

[Voi+20] Paul Voigtlaender, Jonathon Luiten, Philip HS Torr, et al. „Siam r-cnn: Visual tracking by re-detection.“ 2020. [↰ 61, 64]

[WAA17] Wenqi Wang, Vaneet Aggarwal, and Shuchin Aeron. „Efficient low rank tensor ring completion.“ 2017. [↰ 4]

[Wan+17] Jingdong Wang, Ting Zhang, Nicu Sebe, et al. „A survey on learning to hash.“ 2017. [↰ 79]

[Wan+18a] Wenqi Wang, Yifan Sun, Brian Eriksson, et al. „Wide Compression: Tensor Ring Nets.“ 2018. [↰ 4]

[Wan+18b] Xiaolong Wang, Ross Girshick, Abhinav Gupta, et al. „Non-local neural networks.“ 2018. [↰ 52, 54, 55]

[Wan+19] Qiang Wang, Li Zhang, Luca Bertinetto, et al. „Fast online object tracking and segmentation: A unifying approach.“ 2019. [↰ 54]

[Wan+20a] Jingdong Wang, Ke Sun, Tianheng Cheng, et al. „Deep high-resolution representation learning for visual recognition.“ 2020. [↰ 35]

[Wan+20b] Shuohang Wang, Luowei Zhou, Zhe Gan, et al. „Cluster-former: Clustering-based sparse transformer for long-range dependency encoding.“ 2020. [↰ 55]

[Wan+21a] Ning Wang, Wengang Zhou, Jie Wang, et al. „Transformer meets tracker: Exploiting temporal context for robust visual tracking.“ 2021. [↰ 51, 52, 54, 55, 61–64]

[Wan+21b] Wenguan Wang, Tianfei Zhou, Fisher Yu, et al. „Exploring cross-image pixel contrast for semantic segmentation.“ 2021. [↰ 32]

[Wan+21c] Xin Wang, Thomas E Huang, Benlin Liu, et al. „Robust Object Detection via Instance-Level Temporal Cycle Confusion.“ 2021. [↰ 33]

[Wan+21d] Xinlong Wang, Rufeng Zhang, Chunhua Shen, et al. „Dense Contrastive Learning for Self-Supervised Visual Pre-Training.“ 2021. [↰ 31, 32, 35, 38, 49, 103]

[WH18] Yuxin Wu and Kaiming He. „Group normalization.“ 2018. [↰ 36]

[WLY13] Yi Wu, Jongwoo Lim, and Ming-Hsuan Yang. „Online object tracking: A benchmark.“ 2013. [↰ 51, 53, 61, 62, 68, 69]

[WS09] Kilian Q Weinberger and Lawrence K Saul. „Distance metric learning for large margin nearest neighbor classification.“ 2009. [↰ 82]

[Wu+19] Yue Wu, Yinpeng Chen, Lijuan Wang, et al. „Large scale incremental learning.“ 2019. [↰ 103]

[Xie+17] Saining Xie, Ross Girshick, Piotr Dollár, et al. „Aggregated residual transformations for deep neural networks.“ 2017. [↰ 2]

[Xie+20] Qizhe Xie, Minh-Thang Luong, Eduard Hovy, et al. „Self-training with noisy student improves imagenet classification.“ 2020. [↰ 4]

[XT15] Saining Xie and Zhuowen Tu. „Holistically-nested edge detection.“ 2015. [↰ 47]

[Xu+18] Dan Xu, Wanli Ouyang, Xiaogang Wang, et al. „Pad-net: Multi-tasks guided prediction-and-distillation network for simultaneous depth estimation and scene parsing.“ 2018. [↰ 5, 10, 12]

[Yan+18] Tien-Ju Yang, Andrew Howard, Bo Chen, et al. „Netadapt: Platform-aware neural network adaptation for mobile applications.“ 2018. [↰ 4]

[Yan+21a] Bin Yan, Houwen Peng, Jianlong Fu, et al. „Learning spatio-temporal transformer for visual tracking.“ 2021. [↰ 51, 52, 54, 55, 61, 64, 66, 68]

[Yan+21b] Bin Yan, Houwen Peng, Kan Wu, et al. „LightTrack: Finding Lightweight Neural Networks for Object Tracking via One-Shot Architecture Search.“ 2021. [↰ 52, 53, 55, 59–66, 68–73]

[Yan+21c] Bin Yan, Xinyu Zhang, Dong Wang, et al. „Alpha-refine: Boosting tracking performance by precise bounding box estimation.“ 2021. [↰ 54]

[YC18] Tianyu Yang and Antoni B Chan. „Learning dynamic memory networks for object tracking.“ 2018. [↰ 54]

[YK16a] Fisher Yu and Vladlen Koltun. „Multi-scale context aggregation by dilated convolutions.“ 2016. [↰ 2]

[YK16b] Fisher Yu and Vladlen Koltun. „Multi-scale context aggregation by dilated convolutions.“ 2016. [↰ 35]

[YKF17] Fisher Yu, Vladlen Koltun, and Thomas Funkhouser. „Dilated residual networks.“ 2017. [↰ 35]

[Yu+16] Jiahui Yu, Yuning Jiang, Zhangyang Wang, et al. „Unitbox: An advanced object detection network.“ 2016. [↰ 60]

[Yu+18] Fisher Yu, Dequan Wang, Evan Shelhamer, et al. „Deep layer aggregation.“ 2018. [↰ 2]

[Yu+20] Fisher Yu, Haofeng Chen, Xin Wang, et al. „Bdd100k: A diverse driving dataset for heterogeneous multitask learning.“ 2020. [↰ 33, 44]

[Zam+18] Amir R Zamir, Alexander Sax, William Shen, et al. „Taskonomy: Disentangling task transfer learning.“ 2018. [↰ 32]

[Zha+15] Xiangyu Zhang, Jianhua Zou, Kaiming He, et al. „Accelerating very deep convolutional networks for classification and detection.“ 2015. [↰ 4, 13, 18, 80]

[Zha+16] Qibin Zhao, Guoxu Zhou, Shengli Xie, et al. „Tensor ring decomposition.“ 2016. [↰ 13]

[Zha+17] Hengshuang Zhao, Jianping Shi, Xiaojuan Qi, et al. „Pyramid scene parsing network.“ 2017. [↰ 2, 9]

[Zha+18a] Xiangyu Zhang, Xinyu Zhou, Mengxiao Lin, et al. „Shufflenet: An extremely efficient convolutional neural network for mobile devices.“ 2018. [↰ 4, 71, 72]

[Zha+18b] Zhenyu Zhang, Zhen Cui, Chunyan Xu, et al. „Joint task-recursive learning for semantic segmentation and depth estimation.“ 2018. [↰ 10, 12]

[Zha+18c] Xiangyun Zhao, Haoxiang Li, Xiaohui Shen, et al. „A modulation module for multi-task learning with applications in image retrieval.“ 2018. [↰ 11, 12]

[Zha+19a] Xiaohua Zhai, Avital Oliver, Alexander Kolesnikov, et al. „S4l: Self-supervised semi-supervised learning.“ 2019. [↰ 33, 34, 43]

[Zha+19b] Hongyang Zhang, Yaodong Yu, Jiantao Jiao, et al. „Theoretically principled trade-off between robustness and accuracy.“ 2019. [↰ 33]

[Zha+19c] Lichao Zhang, Abel Gonzalez-Garcia, Joost van de Weijer, et al. „Learning the model update for siamese trackers.“ 2019. [↰ 54]

[Zha+19d] Zhenyu Zhang, Zhen Cui, Chunyan Xu, et al. „Pattern-Affinitive Propagation across Depth, Surface Normal and Semantic Segmentation.“ 2019. [↰ 10, 12]

[Zha+20] Zhipeng Zhang, Houwen Peng, Jianlong Fu, et al. „Ocean: Object-aware anchor-free tracking.“ 2020. [↰ 60, 61, 66, 69]

[Zha+22] Xiaohua Zhai, Alexander Kolesnikov, Neil Houlsby, et al. „Scaling vision transformers.“ 2022. [↰ 2]

[Zha+95] Zhengyou Zhang, Rachid Deriche, Olivier Faugeras, et al. „A robust technique for matching two uncalibrated images through the recovery of the unknown epipolar geometry.“ 1995. [↰ 1]

[Zho+17] Tinghui Zhou, Matthew Brown, Noah Snavely, et al. „Unsupervised learning of depth and ego-motion from video.“ 2017. [↰ 2]

[Zhu+17] Chenzhuo Zhu, Song Han, Huizi Mao, et al. „Trained ternary quantization.“ 2017. [↰ 4]

[Zhu+19] Bohan Zhuang, Chunhua Shen, Mingkui Tan, et al. „Structured binary neural networks for accurate image classification and semantic segmentation.“ 2019. [↰ 87]

[ZK16] Sergey Zagoruyko and Nikos Komodakis. „Wide Residual Networks.“ 2016. [↰ 2]

[Zop+18] Barret Zoph, Vijay Vasudevan, Jonathon Shlens, et al. „Learning transferable architectures for scalable image recognition.“ 2018. [↰ 4]

[ZY21] Qinglong Zhang and Yu-Bin Yang. „ResT: An efficient transformer for visual recognition.“ 2021. [↰ 55]

CURRICULUM VITAE

Name	Menelaos Kanakis
Date of Birth	8 September, 1991
Nationality	Cyprus, United Kingdom

EDUCATION

2019 – 2023 **Doctor of Sciences of ETH Zürich**
ETH Zürich, Department of Electrical Engineering and Information Technology (D-ITET)
Computer Vision Laboratory
Supervisor: Prof. Luc Van Gool
Title: Designing Efficient Deep Neural Networks

2016 – 2018 **Master of Science in Robotics, Systems and Control**
ETH Zürich
Grade: 5.53/6

2012 – 2016 **Bachelor of Engineering (with Honours) Mechanical Engineering (with a year in industry)**
University of Leeds
Grade: First Class Honours

PROFESSIONAL EXPERIENCE

2019 – 2023 **Scientific Assistant, ETH Zürich**
Computer Vision Laboratory
Publications, student supervision, and academic service

2017 – 2018 **Reserach Assistant, ETH Zürich**
Product Development Group
Publications, and student supervision

2017 – 2018 **Applications Engineer, National Instruments**
Technical support, course teaching, and development of demos for marketing purposes

2014 – 2014 **Research and Development Engineer, SignalGeneriX LTD**
Assist the R&D team to design and develop innovative products

PUBLICATIONS

[1] Menelaos Kanakis, Thomas E Huang, David Bruggemann, Fisher Yu, Luc Van Gool, "Composite Learning for Robust and Effective Dense Predictions", WACV, 2023.

[2] Philippe Blatter*, Menelaos Kanakis*, Martin Danelljan, Luc Van Gool, "Efficient Visual Tracking with Exemplar Transformers", WACV, 2023.

[3] Menelaos Kanakis*, Simon Maurer*, Matteo Spallanzani, Ajad Chhatkuli, Luc Van Gool, "ZippyPoint: Fast Interest Point Detection, Description, and Matching through Mixed Precision Discretization", arXiv preprint arXiv:2203.03610, 2022.

[4] David Bruggemann, Menelaos Kanakis, Anton Obukhov, Stamatios Georgoulis, Luc Van Gool, "Exploring Relational Context for Multi-Task Dense Prediction", ICCV, 2021.

[5] Guolei Sun, Thomas Probst, Danda Pani Paudel, Nikola Popovic, Menelaos Kanakis, Jagruti Patel, Dengxin Dai, Luc Van Gool, "Task Switching Network for Multi-Task Learning", ICCV, 2021.

[6] Suman Saha*, Anton Obukhov*, Danda Pani Paudel, Menelaos Kanakis, Yuhua Chen, Stamatios Georgoulis, Luc Van Gool, "Learning to Relate Depth and Semantics for Unsupervised Domain Adaptation", CVPR, 2021.

[7] David Bruggemann, Menelaos Kanakis, Stamatios Georgoulis, Luc Van Gool, "Automated Search for Resource-Efficient Branched Multi-Task Networks", BMVC, 2020.

[8] Menelaos Kanakis, David Bruggemann, Suman Saha, Stamatios Georgoulis, Anton Obukhov, Luc Van Gool, "Reparameterizing Convolutions for Incremental Multi-Task Learning without Task Interference", ECCV, 2020.

[9] Anton Obukhov, Maxim Rakhuba, Stamatios Georgoulis, Menelaos Kanakis, Dengxin Dai, Luc Van Gool, "T-Basis: a Compact Representation for Neural Networks", ICML, 2020.

* Equal contribution.

[10] Suman Saha, Wenhao Xu, Menelaos Kanakis, Stamatios Georgoulis, Yuhua Chen, Danda Pani Paudel, Luc Van Gool, "Domain Agnostic Feature Learning for Image and Video Based Face Anti-spoofing", CVPRW, 2020.

PATENTS

2021 (pending) "Method and system for compressing a neural network." European Patent 3923199A1.

ACADEMIC SERVICE

2023 **Conference reviewer**
CVPR, WACV

2022 **Conference reviewer**
CVPR, ECCV

2021 **Conference reviewer**
CVPR, ICCV

2020 **Conference reviewer**
CVPR