KU LEUVEN

ARENBERG DOCTORAL SCHOOL
Faculty of Engineering Science

Multi-Task Learning for Visual Scene Understanding

Simon Vandenhende

Supervisor:
Prof. Luc Van Gool

Dissertation presented in partial fulfillment of the requirements for the degree of Doctor of Engineering Science (PhD): Electrical Engineering

March 2022

Multi-Task Learning for Visual Scene Understanding

Simon VANDENHENDE

Examination committee:
Prof. Paul Sas, chair
Prof. Luc Van Gool, supervisor
Prof. Tinne Tuytelaars
Prof. Marie-Francine Moens
Prof. Matthew Blaschko

Prof. Hakan Bilen
(University of Edinburgh)
Prof. Gemma Roig
(Goethe University)

Dissertation presented in partial fulfillment of the requirements for the degree of Doctor of Engineering Science (PhD): Electrical Engineering

March 2022

© 2022 KU Leuven – Faculty of Engineering Science
Uitgegeven in eigen beheer, Simon Vandenhende, Kasteelpark Arenberg 10 - box 2441, B-3001 Leuven (Belgium)

Alle rechten voorbehouden. Niets uit deze uitgave mag worden vermenigvuldigd en/of openbaar gemaakt worden door middel van druk, fotokopie, microfilm, elektronisch of op welke andere wijze ook zonder voorafgaande schriftelijke toestemming van de uitgever.

All rights reserved. No part of the publication may be reproduced in any form by print, photoprint, microfilm, electronic or any other means without written permission from the publisher.

Preface

A PhD is more than just another university degree. A PhD is about the pursuit of knowledge, the aspiration to figure things out and the love to explore new ideas. It is a personal journey filled with challenges, discoveries and encounters with extraordinary people. As I reflect on this incredible adventure, a feeling of great contentment prevails. Now all that rests is to thank my collaborators, colleagues, friends and family members for helping me to achieve my goals.

I wish to start by thanking my advisor Luc Van Gool for taking me on board as one of his PhD students. Luc, thank you for our discussions, for putting your trust in me even when things did not go according to plan and for all the freedom in research you provided. You created a unique place where I could freely pursue my research interests without distractions. The environment you provided allowed me to turn my ideas into reality and push my research further than what I hoped for. Despite all your accomplishments in research, you were always down-to-earth and ready to hear out my opinion. I am most honored that you chose to advise me.

I am thankful for the members of my supervisory committee Prof. Tinne Tuytelaars, Prof. Marie-Francine Moens and Prof. Matthew Blaschko for providing me with feedback and support throughout the years. Further, I would like to thank Prof. Hakan Bilen and Prof. Gemma Roig for promptly agreeing to serve on my examination committee. Also, I thank Prof. Paul Sas for serving as a chair.

This manuscript would have never been in its current state without the guidance of Dr. Stamatios Georgoulis, who took me under his wings when I started my PhD. Stam, thank you for acting as a mentor, for believing in me, for having countless meetings with me, for sharing your knowledge with me, and for teaching me how to write papers. It is hard to put into words the gratitude that I feel for the endless support you gave me. I hope we can meet again soon!

Another remarkable colleague was Wouter Van Gansbeke, who passed on to me his interest for self-supervised learning. Wouter, I am very proud of the projects we did together. You are a very good researcher and a dedicated worker. I am thankful for our

collaborations throughout the years!

I would also like to thank my other co-authors, Thierry Deruyttere, Dusan Grujicic, Davy Neven, Bert De Brabandere and Marc Proesmans. Thank you Thierry and Dusan for being great colleagues, for working long hours when we had a deadline and for all the fun discussions. Davy and Bert, it was nice to have you as my master thesis supervisors. I am happy we became colleagues and friends after! Marc, thank you for your endless feedback, for being a great listener and for all the times you joined me for a coffee break.

Further, I was also lucky to be surrounded by awesome office mates, Yu-Hui Huang, Yevhen Kuznietsov, Wouter Van Gansbeke and Liqian Ma. Our office certainly did not qualify as a standard working place as there were many distractions. Rare moments of quiet concentration were often interrupted by Yevhen, who wanted to share his opinion on something. I will certainly miss the fun moments we shared in the office. I would also like to thank Ali Diba - my office mate during my early days at Visics. Ali, thanks for passing on your knowledge and for all the fun talks we had!

Some of the nicer things about the PhD were the trips abroad - where we had some time to relax. I fondly remember the trip to Tokyo with Yu-Hui and Wouter - where we did plenty of sight-seeing, eating sushi and walking during our free time. Another great trip was the summer school in Sicily with Matthias, Thomas, Yevhen and Wouter. The lectures were alternated with swimming, playing volleyball and drinking Sicilian wine.

I am also grateful for my other colleagues at Visics and Trace, who turn this place into a nice working environment: Matthias, Ali (2x), Dusan, Yu-Hui, Abhishek, Yevhen, Tim, Liqian, Xingchen, Nicola, Robby, Cedric, Gorjan, Soroush, Wouter, Thomas, Eli, Kaili, Bert (2x), Davy, Rahaf, Maxim, Klaas, Marc, Gioa, Bruno, Wim, Jonas, Michael, Mark, Aymen, Paul, Pattrica, Annitta,... Thanks for all the birthday cakes, the volleyball games, the beers at Oude Markt, etc.

Finally, I reserve this part to thank some other important people in my life. My achievements would be both impossible and meaningless without them. Thank you to all my relatives and friends, the ones in Belgium, and the ones abroad. A special thanks goes to my friends from Roeselare: Arnaud, Dieter, Gilbert, Giel, Guillaume, Jan, Jens, Jos, Kenneth, Robin and Sam. I am very grateful for our friendship! A big thanks also goes to the people that turned Leuven into a wonderful place to live. I am thankful for all the people who became my friends during my time in Leuven. I won't forget our nights out! I would also like to thank Malou for bringing so much joy and happiness into my life. Thank you for always being there! Finally, the most special thanks goes to my parents - Stefaan and Carine - my brother - Maarten - and my other family members for their unconditional love and support.

Abstract

Despite the recent progress in deep learning, most approaches still go for a silo-like solution, training a separate neural network for each individual task. Yet, many real-world problems require to solve a multitude of tasks concurrently. For example, an autonomous vehicle should be able to detect all objects in the scene, localize them, estimate their distance and trajectory, etc., in order to safely navigate itself in its surroundings. Similarly, an image recognition system for commerce applications should be able to tag products, retrieve similar items, come up with personalized recommendations, etc., to provide customers with the best possible service. These types of problems have motivated researchers to build multi-task learning models.

The key idea of multi-task learning is to learn multiple tasks in parallel while sharing the learned representations. Multi-task networks bring several practical advantages compared to the single-tasking case, where each individual task is solved separately by its own network. First, due to the sharing of layers, the resulting memory footprint is substantially reduced. Second, as they avoid to repeatedly calculate features in the shared layers, once for every task, they show increased inference speeds. Third, they have the potential to improve the performance if the associated tasks share complementary information, or act as a regularizer for one another.

We are faced with two important challenges when building multi-task learning models. First, we need to come up with neural network architectures that can handle multiple tasks. Second, we need to develop novel training recipes for jointly learning the tasks. In particular, since we are optimizing for multiple objectives in parallel, one or more tasks could start to dominate the weight update process, hindering the model from learning the other tasks. In this manuscript, we delve into both problems within the context of visual scene understanding.

We propose two new types of models to address the architectural problem. First, we explore branched multi-task networks, where the deeper layers of the neural network gradually grow more task-specific. We introduce a principled method to automate the construction of such branched multi-task networks. The construction process groups

together the tasks that can be solved with a similar set of features, while making a trade-off between task similarity and network complexity. In this way, our method produces models that yield a better trade-off between the performance and the amount of computational resources.

Second, we propose a novel neural network architecture for jointly tackling multiple dense prediction tasks. The key idea is to improve the predictions for each task by distilling useful information from the predictions of other tasks at *multiple scales*. The motivation for including multiple scales is based upon the observation that tasks with high affinity at a certain scale are not guaranteed to retain this behavior at other scales, and vice versa. Extensive experiments across two popular benchmarks for dense labeling show that, unlike prior work, our model delivers on the full potential of multi-task learning, that is, smaller memory footprint, reduced number of calculations, and better performance w.r.t. single-task learning.

Further, we also consider the multi-task learning optimization problem. We first analyze several existing techniques for balancing the learning of tasks. Surprisingly, we uncover several discrepancies between works. We hypothesize that this can be attributed to the lack of standardized benchmarks for multi-task learning, and different benchmarks benefit from a specific strategy. Based upon this result, we then isolate the elements that are most promising and propose a set of heuristics to balance the tasks. The heuristics are of practical nature, and result in more robust performance across different benchmarks.

In the final chapter, we consider the problem of scene understanding from an alternative point-of-view. Many of the models described in the literature benefit from supervised pretraining. In this case, the model is first pretrained on a larger annotated dataset like ImageNet, before being transferred to the tasks of interest. This allows the model to perform well, even on datasets with only a modest amount of labeled examples. Unfortunately, supervised pretraining relies on an annotated dataset itself, which limits it's applicability. To address this problem, researchers started to explore self-supervised learning methods. We revisit recent popular works based upon contrastive learning. First, we show that an existing method like MoCo obtains robust results across different datasets, including scene-centric, long-tailed and domain-specific data. Second, we improve the learned representations by imposing additional invariances. This result directly benefits many downstream tasks like semantic segmentation, detection, etc. Finally, we show that the improvements obtained via self-supervised learning translate to multi-task learning networks too.

In summary, this manuscript presents several important contributions that improve multi-task learning models for visual scene understanding. The innovations focus on improving the neural network architecture, the optimization process and the pretraining aspect. All methods are thoroughly tested across a variety of benchmarks. The code is made publicly available: `https://github.com/SimonVandenhende`.

Beknopte samenvatting

Ondanks de recente vooruitgang in *machine learning*, gesterkt door het gebruik van diepe neurale netwerken, worden modellen nog steeds gebouwd voor individuele taken. Hierbij wordt een apart neuraal netwerk getraind voor elke taak. Vele praktische problemen vereisen echter dat we verschillende taken synchroon uitvoeren. Om bijvoorbeeld veilig te kunnen navigeren moet een autonoom voertuig gelijktijdig alle objecten in de scène detecteren, hun locatie bepalen, de afstand schatten, etc. Op gelijkaardige wijze moet een intelligent systeem voor commerciële applicaties producten herkennen, gelijkaardige items ophalen, persoonlijke aanbiedingen voorstellen, etc. Dit om klanten de beste mogelijke ervaring te bieden. Om dergelijke problemen op te lossen begonnen onderzoekers aan de de bouw van *multi-task learning* modellen die meerdere taken gelijktijdig kunnen uitvoeren.

Het idee achter *multi-task learning* is om meerdere taken in parallel te leren, terwijl de geleerde representaties gedeeld worden tussen taken. In vergelijking met het afzonderlijk afhandelen van elke taak bieden *multi-task learning networks* verschillende praktische voordelen. Ten eerste laat het delen van de representaties toe om minder computergeheugen te gebruiken. Verder kan een deel van de berekeningen afgeschreven worden over meerdere taken, wat de totale hoeveelheid rekenwerk vermindert. Tenslotte verbetert ook de performantie wanneer taken informatie kunnen uitwisselen.

Er zijn een aantal uitdagingen voor het bouwen van *multi-task learning* modellen. Allereerst moeten we nieuwe neurale netwerkarchitecturen bedenken die geschikt zijn voor het behandelen van meerdere taken. Ten tweede moeten we optimalisatietechnieken ontwikkelen die toelaten om het model meerdere taken in parallel aan te leren. Een uitdaging hierbij is dat een enkele of meerdere taken het leerproces kunnen domineren en het model bijgevolg niet in staat is om de resterende taken te leren. In deze thesis behandelen we beide uitdagingen binnen de context van *visual scene understanding.*

We stellen twee nieuwe neurale netwerk architecturen voor. Ten eerste bestuderen we *branched multi-task networks*, waar de diepere lagen van het neurale netwerk gradueel meer taakspecifiek worden. We introduceren een nieuw algoritme voor de

automatische constructie van dergelijke netwerken. Het algoritme groepeert taken die opgelost kunnen worden aan de hand van een gelijkaardige verzameling visuele kenmerken. Tegelijkertijd brengen we ook de complexiteit van het neurale netwerk in rekening. De modellen gegenereerd met onze methode realiseren een betere trade-off tussen de performantie en de benodigde rekenkracht. Dit wordt duidelijk aangetoond in onze experimentele resultaten.

Het tweede model dat we voorstellen focust expliciet op meerdere per-pixel-taken. Het idee is om de predicties voor elke taak te verbeteren door informatie te halen uit de predicties van andere taken. We passen dit principe toe op meerdere schalen nadat we via observatie hebben geconstateerd dat de geometrische relaties tussen taken wijzigen afhankelijk van de gebruikte schaal. Op deze wijze halen we de maximale hoeveelheid informatie uit andere taken voor het verbeteren van de predicties. De experimentele resultaten tonen aan dat ons model beter presteert met betrekking tot de hoeveelheid computergeheugen, het aantal bewerkingen en de performantie.

We onderzoeken ook het *multi-task-learning-optimalisatieprobleem*. Eerst voeren we een vergelijkende studie uit van bestaande werken. Tot onze verbazing ontdekken we verschillende discrepanties tussen bestaande werken. De verklaring hiervoor is dat de methodes ontwikkeld werden voor een specifieke dataset of toepassing. Gebaseerd op dit resultaat proberen we de meest veelbelovende elementen te isoleren en stellen we een aantal heuristieken voor. De heuristieken zijn praktisch georiënteerd. Verder tonen we aan dat het gebruik van heuristieken zich vertaalt in meer robuuste resultaten.

In het laatste hoofdstuk onderzoeken we het probleem van *visual scene understanding* vanuit een alternatief oogpunt. Veel modellen maken gebruik van gesuperviseerd pretrainen. Hierbij wordt het model eerst getraind op een andere, grotere dataset, waarna de gewichten overgedragen worden naar de taak waarin we geïnteresseerd zijn. Dit laat toe om een hoge performantie te behalen, zelfs op datasets met slechts een beperkte hoeveelheid gelabelde voorbeelden. Gesuperviseerd pretrainen maakt echter zelf gebruik van een gelabelde dataset en dit beperkt het gebruik. Onderzoekers werken daarom op zelf-gesuperviseerde methodes, waarbij het leersignaal vanuit de data zelf komt. We bestuderen recente methodes gebaseerd op *contrastive learning*. Eerst tonen we aan dat een bestaande methode zoals *MoCo* robuuste resultaten kan behalen voor verscheidene datasets - waaronder *scene-centric*, ongebalanceerde en domeinspecifieke datasets. Verder verbeteren we de representaties door extra invarianties op te leggen. Dit resultaat laat toe bestaande modellen voor segmentatie, objectdetectie, etc. te verbeteren. Tenslotte tonen we aan dat we op een gelijkaardige manier ook de performantie van *multi-task learning* modellen kunnen verbeteren.

Deze thesis beschrijft verschillende methodes voor het verbeteren van *multi-task learning* modellen voor *visual scene understanding*. De contributies focussen op het verbeteren van de netwerkarchitectuur, het optimalisatieproces en het pre-trainingsaspect. Alle innovaties werden getest op verschillende datasets. De bijhorende

code is publiek beschikbaar: `https://github.com/SimonVandenhende`.

Contents

Abstract iii

Beknopte samenvatting v

Contents ix

List of Figures xiii

List of Tables xv

1 Introduction 1
- 1.1 Challenges . . . 2
- 1.2 Overview & Contributions . . . 4

2 Background 7
- 2.1 Multi-Task Learning in Neural Networks . . . 7
- 2.2 Datasets . . . 9
- 2.3 Evaluation Criteria . . . 10
- 2.4 Conclusion . . . 12

3 Related Work 13
- 3.1 Deep Multi-Task Architectures . . . 14
  - 3.1.1 Historical Overview and Taxonomy . . . 15
  - 3.1.2 Encoder-focused Architectures . . . 18
  - 3.1.3 Decoder-Focused Architectures . . . 20
  - 3.1.4 Other Approaches . . . 24
- 3.2 Optimization in MTL . . . 25
  - 3.2.1 Task Balancing Approaches . . . 25
  - 3.2.2 Other Approaches . . . 30
- 3.3 Conclusion . . . 31

4 Branched Multi-Task Networks **33**
4.1 Introduction . . . 34
4.2 Related Work . . . 35
4.3 Method . . . 36
4.3.1 Calculate Task Affinity Scores . . . 37
4.3.2 Construct a Branched Multi-Task Network . . . 38
4.4 Experiments . . . 39
4.4.1 Cityscapes . . . 39
4.4.2 Taskonomy . . . 41
4.4.3 CelebA . . . 43
4.4.4 Computational Analysis . . . 45
4.5 Conclusion . . . 46

5 Multi-Scale Multi-Task Interaction Networks **47**
5.1 Motivation . . . 48
5.1.1 Decoder-Focused Approaches . . . 48
5.1.2 Task interactions at different scales . . . 49
5.2 Method . . . 50
5.2.1 Multi-scale multi-modal distillation . . . 50
5.2.2 Feature propagation across scales . . . 52
5.2.3 Feature aggregation . . . 53
5.2.4 Optimization . . . 54
5.3 Experiments . . . 54
5.3.1 Setup . . . 54
5.3.2 Ablation Studies . . . 55
5.3.3 State-Of-The-Art Comparison . . . 58
5.4 Conclusion . . . 61

6 Task Balancing Strategies **63**
6.1 Introduction . . . 63
6.2 Qualitative Comparison . . . 64
6.3 Experiments . . . 65
6.3.1 Setup . . . 65
6.3.2 Results . . . 67
6.3.3 Discussion . . . 71
6.4 Conclusion . . . 72

7 Self-Supervised Learning **73**
7.1 Introduction . . . 74
7.2 Framework . . . 76
7.3 Contrastive Learning in the Wild . . . 76
7.3.1 Setup . . . 77
7.3.2 Object-centric Versus Scene-centric . . . 80

7.3.3 Uniform Versus Long-tailed 81
7.3.4 Domain-Specific Datasets 81
7.4 Learning Invariances 82
7.4.1 Multi-Scale Constrained Cropping 82
7.4.2 Stronger Augmentations 85
7.4.3 Nearest Neighbors 89
7.4.4 Discussion 93
7.5 Related Work 93
7.6 Application: Indoor Scene Understanding 94
7.7 Conclusion 95

8 Conclusion **97**
8.1 Summary of Contributions 97
8.2 Limitations and Future Work 99

Curriculum **115**

List of Publications **117**

List of Figures

2.1 The single-task learning setup 8
2.2 The multi-task learning setup 8
2.3 Dataset examples 11

3.1 A taxonomy of multi-task learning approaches for computer vision . . 13
3.2 Soft versus hard parameter sharing models 14
3.3 Encoder- versus decoder-focused models 17
3.4 Cross-stitch networks and NDDR-CNNs 18
3.5 Multi-Task Attention Networks 18
3.6 PAD-Net 21
3.7 Pattern-Affinitive Propagation Networks 23
3.8 Joint-Task Recursive Learning 24

4.1 Method overview of branched multi-task networks 36
4.2 Task relatedness measured on Cityscapes 40
4.3 Results for branched multi-task networks on Cityscapes 40
4.4 Task groupings on Cityscapes 41
4.5 Task groupings on Taskonomy 43
4.6 Task groupings on CelebA 45

5.1 PAD-Net 48
5.2 Task interactions change with scale - qualitative analysis 50
5.3 Task interactions change with scale - quantitative analysis 50
5.4 MTI-Net 51
5.5 Feature Propagation Module in MTI-Net 53
5.6 Qualitative results on PASCAL 57
5.7 Qualitative results on NYUD 61

6.1 Performance profiles of different task balancing strategies 70

7.1 Example images from the pretraining datasets. 78

7.2 IoU between random resized crops in self-supervised contrastive frameworks 81
7.3 Transfer results when limiting the IoU between crops in self-supervised contrastive frameworks 81
7.4 Training curves when limiting the IoU between crops in self-supervised contrastive frameworks 81
7.5 Visualization of the constrained multi-crop transform 84
7.6 Transfer performance versus time for the multi-crop transform 84
7.7 Class activation maps for the two-crop versus multi-crop model. . . . 86
7.8 Overview of the kNN-MoCo setup 91

List of Tables

2.1 Overview of the multi-task learning benchmarks. 10

4.1 Results for branched multi-task networks on Cityscapes 41
4.2 Results for branched multi-task networks on Taskonomy 42
4.3 Results for branched multi-task networks on CelebA 44

5.1 Ablation study of MTI-Net 56
5.2 Ablation study of the number of scales 58
5.3 Ablation study of the information flow in MTI-Net 58
5.4 State-of-the-art comparison on PASCAL 59
5.5 State-of-the-art comparison on NYUD 60
5.6 Computational resource analysis 62

6.1 Qualitative comparison of task balancing techniques 66
6.2 Experimental comparison of task balancing techniques 68

7.1 Overview of the pretraining datasets 78
7.2 Comparison of linear classification models using different pretraining datasets 79
7.3 Comparison of representations under the transfer learning setup when using different pretraining datasets 79
7.4 Ablation study of learning additional invariances 83
7.5 Results for the DAVIS 2017 video instance segmentation task 87
7.6 Results for the PASCAL VOC semantic segment retrieval task 88
7.7 Ablation of augmentation policies applied to random crops. 89
7.8 Ablation study of the number of neighbors k and weight $lambda$ for kNN-MoCo 91
7.9 State-of-the-art comparison between self-supervised pretraining methods on COCO 92

7.10 Comparison of pretraining strategies under a single- and multi-task learning setup on NYUD. We consider the tasks of semantic segmentation and monocular depth estimation. The multi-task learning model with MoCo pretrained weights reports the best results on both tasks. 95

Chapter 1

Introduction

Machine learning has become a ubiquitous technology - it's applications range from making recommendations on e-commerce websites to recognizing crop types in agriculture. Machine learning algorithms build models based on training examples, in order to make predictions or decisions for new samples, avoiding the need to program the desired behavior explicitly. Many applications that use machine learning rely on a class of techniques named deep learning.

Deep learning [49] uses models that consist of multiple processing layers to learn representations of data with multiple levels of abstraction. The representations automatically discover relevant structures in the data through the backpropagation algorithm, which tells how the internal parameters of the representations should be adjusted. This setup diverges from prior attempts that involved the use of handcrafted features. Deep learning has been successfully applied in text processing [124], speech recognition [3] and visual object recognition [59].

In computer vision, deep neural networks have reported state-of-the-art-results for semantic segmentation [104], instance segmentation [57] and depth estimation [42]. Traditionally, these tasks are tackled in isolation, meaning a separate neural network is trained for each task. Yet, many real-world problems require to solve a multitude of tasks concurrently. For example, an autonomous vehicle should be able to detect all objects in the scene, localize them, estimate their distance and trajectory, etc., in order to safely navigate itself in it's surroundings. Similarly, an image recognition system for e-commerce applications should be able to recognize products, retrieve similar items, provide personalized content, etc., to provide customers with the best possible service. These problems have motivated researchers to develop multi-task learning models that jointly infer all desired task outputs.

There is a biological argument for building multi-task learning models as well. Humans are remarkably good at solving many different tasks. It appears that biological data processing follows a multi-task learning strategy too: instead of separating tasks and tackling them in isolation, different processes seem to share the same early processing layers in the brain (e.g. V1 in macaques [53]).

In the deep learning era, multi-task learning [17, 153] (MTL) translates to designing networks capable of learning shared representations from multi-task supervisory signals. Compared to the single-task case, where each individual task is solved separately by its own networks, such multi-task networks bring several advantages to the table. First, due to their inherent layer sharing, the resulting memory footprint is substantially reduced. Second, as they explicitly avoid to repeatedly calculate features in the shared layers, once for every task, they show increased inference speeds. Third, they have the potential for improved performance if the associated tasks share complementary information, or act as a regularizer for one another.

In this thesis, we study the problem of multi-task learning within the context of visual scene understanding. We develop new techniques - including better model architectures and training recipes - that improve over the current state-of-the-art for jointly tackling multiple visual recognition tasks. The proposed methods have the potential to directly benefit the many practical applications that rely on models for visual scene understanding.

1.1 Challenges

We identify the most important challenges for jointly learning multiple visual recognition tasks that we tackle in this thesis.

Network Architecture We need to develop novel neural network architectures that are capable of handling multiple tasks. Early attempts used either hard or soft parameter sharing for learning multiple tasks. In hard parameter sharing [73, 77], the input is first encoded through a shared network of layers after which tasks branch out into their own sequence of task-specific layers. Differently, in soft parameter sharing [114, 47], a set of task-specific networks is used in combination with a feature sharing mechanism. For example, Misra et al. [114] proposed to use cross-stitch units that take a linear combination of the features in the intermediate layers of the network to fuse information.

The aforementioned works followed a common pattern: they directly predict all task outputs from the same input in one processing cycle. In particular, the predictions are generated once, in parallel or sequentially, and are not refined afterwards. By

doing so, these works fail to capture commonalities and differences among tasks, that are likely fruitful for one another (e.g. depth discontinuities are usually aligned with semantic edges). To alleviate this issue, a few recent works [165, 178] first employed a multi-task network to make initial task predictions, and then leveraged features from these initial predictions in order to further improve each task output - in a one-off or recursive manner.

These different methods each have their own advantages and disadvantages. Therefore, the main challenge will be to find elegant solutions that address the shortcomings of existing methods. Also, note that there are several aspects that need to be taken into account. We do not wish to optimize for performance alone, but would also like to consider the memory footprint and the number of computations that are needed. Often, there will be a trade-off between the performance and computational resources that are required. As a result, the method of choice can depend on the available computational resources.

Optimization Another important challenge of jointly learning multiple tasks is properly weighting the loss functions associated with the individual tasks. In particular, the network weight update can be suboptimal when the task gradients conflict, or dominated by one task when its gradient magnitude is much higher with respect to the other tasks.

There have been several attempts already to address this problem. Early work [73] used the homoscedastic uncertainty of each task to weigh the losses. Gradient normalization [27] balances the learning of tasks by dynamically adapting the gradient magnitudes in the network. Liu *et al.* [101] weigh the losses to match the pace at which different tasks are learned. Dynamic task prioritization [51] prioritizes the learning of difficult tasks. Sener *et al.* [139] cast multi-task learning as a multi-objective optimization problem, with the overall objective of finding a Pareto optimal solution.

The availability of all these different methods might give the false impression that the problem of setting the task-specific weights is solved. However, as we will show in this manuscript, many existing methods were designed with specific datasets or tasks in mind. In contrast, we aim to develop techniques that are more robust across different settings.

Pretraining Existing models for scene understanding often rely on supervised pretraining to obtain good performance. This builds upon the fact that the weights of convolutional neural networks transfer well between tasks. In this way, pretraining on a large dataset like ImageNet [33] allows to perform well on tasks with only a modest amount of labelled data. This is the case for both single- and multi-task learning networks.

Unfortunately, supervised pretraining requires access to an annotated pretraining dataset which limits it's applicability. To resolve this problem, researchers have begun to explore self-supervised learning methods which learn representations by solving pretext tasks for which the supervisory signal comes for free. Self-supervised learning can leverage huge amounts of raw unlabeled images.

Although the pretraining aspect is not directly tied to multi-task learning itself, we study it's use as it proves a crucial component for existing scene understanding models described in the literature. Also, an interesting property of working on the pretraining aspect, is that any improvements directly translate to several downstream tasks like object detection, semantic segmentation, etc.

1.2 Overview & Contributions

This manuscript focuses on multi-task learning with deep neural networks in the context of visual scene understanding. The work presented in this thesis has been spread across several academic papers. We present an overview and list the contributions made in each chapter.

Chapter 2 familiarizes the reader with the necessary background materials. We define the multi-task learning setup within the context of deep neural networks. Further, we introduce the datasets and evaluation metrics that are used throughout this thesis.

Chapter 3 reviews recent methods for multi-task learning, including both architectural and optimization based strategies. For each method, we describe its key aspects, discuss the commonalities and differences with related works, and present the possible advantages or disadvantages. Further, we also provide a novel taxonomy to classify recent contributions to deep multi-task learning within the context of computer vision. Finally, starting from our literature review, we motivate the various lenses through which we study the multi-task learning problem in the chapters that follow. The content of this chapter is based upon the following publication (journal):

- ***Vandenhende, S.**, Georgoulis, S., Van Gansbeke, W., Proesmans, M., Dai, D., & Van Gool, L. (2021). Multi-task learning for dense prediction tasks: A survey. IEEE Transactions on Pattern Analysis and Machine Intelligence. [153]*

Chapter 4 presents a novel architecture for multi-task learning. We study branched multi-task networks, where layers gradually grow more task-specific as we move to the deeper layers. Further, we introduce a principled approach to automatically construct branched multi-task networks. The construction process groups related tasks together while striking a trade-off between task similarity and network complexity. The task

relatedness scores are based on the premise that similar tasks can be solved with similar features. Our experimental evaluation shows that our architectures outperform other competing models which require a comparable amount of parameters or computations. The content of this chapter is based upon the following publication (poster presentation):

- ***Vandenhende, S.**, Georgoulis, S., De Brabandere, B., & Van Gool, L. (2020). Branched Multi-Task Networks: Deciding What Layers To Share. The British Machine Vision Conference. [152]*

Chapter 5 explicitly focuses on devising a better architecture for tackling multiple pixel-level prediction tasks. We argue about the importance of modeling task interactions at multiple scales when distilling the task information under a multi-task learning setup. In contrast to common belief, we show that tasks with high affinity at a certain scale are not guaranteed to retain this behaviour at other scales, and vice versa. We propose a novel architecture, namely MTI-Net, that builds upon this finding in three ways. First, it explicitly models task interactions at every scale via a multi-scale multi-modal distillation unit. Second, it propagates distilled task information from lower to higher scales via a feature propagation module. Third, it aggregates the refined task features from all scales via a feature aggregation unit to produce the final per-task predictions. Experiments on two multi-task dense labeling datasets show that our model delivers on the full potential of multi-task learning, that is, smaller memory footprint, reduced number of calculations, and better performance w.r.t single-task learning. The content of this chapter is based upon the following publication (spotlight presentation):

- ***Vandenhende, S.**, Georgoulis, S., & Van Gool, L. (2020). MTI-Net: Multi-scale task interaction networks for multi-task learning. European Conference on Computer Vision.*

Chapter 6 considers the problem of balancing the learning of tasks. First, we analyze existing strategies and isolate the elements that are most successful. We observe that existing methods still suffer from several shortcomings and highlight the discrepancies between existing works. Finally, we propose a set of heuristics to balance the tasks. The use of heuristics results in more robust performance across different datasets. The content of this chapter is based upon the following publication (journal):

- ***Vandenhende, S.**, Georgoulis, S., Van Gansbeke, W., Proesmans, M., Dai, D., & Van Gool, L. (2021). Multi-task learning for dense prediction tasks: A survey. IEEE Transactions on Pattern Analysis and Machine Intelligence. [153]*

Chapter 7 studies contrastive self-supervised learning. First, we study how biases in the dataset affect an existing method like MoCo. We empirically show that an

approach like MoCo works well across: (i) object- versus scene-centric, (ii) uniform versus long-tailed and (iii) general versus domain-specific datasets. Second, given the generality of the approach, we try to realize further gains with minor modifications. We show that learning additional invariances - through the use of multi-scale cropping, stronger augmentations and nearest neighbors - improves the representations. Further, we observe that MoCo learns spatially structured representations when trained with a multi-crop strategy. The representations can be used for semantic segment retrieval and video instance segmentation without finetuning. Moreover, the results are on par with specialized models. Finally, we show that MoCo can be used to improve an existing model for indoor scene understanding. The content of this chapter is based upon the following publication [1]:

- *Van Gansbeke, W.*, **Vandenhende, S.***, Georgoulis, S., & Van Gool, L. (2021). Revisiting Contrastive Methods for Unsupervised Learning of Visual Representations. NeurIPS.*

Chapter 8 concludes this manuscript.

Acknowledgment. I would like to acknowledge support by Toyota via the TRACE project and MACCHINA (KU Leuven, C14/18/065). This work is also sponsored by the Flemish Government under the Flemish AI programme.

[1] * denotes equal contribution.

Chapter 2

Background

This chapter introduces the necessary background materials related to our work. First, we briefly discuss how neural networks are utilized under the multi-task learning setup. Second, we cover the different datasets that will be used in this manuscript. Third, we describe our evaluation criteria.

2.1 Multi-Task Learning in Neural Networks

In machine learning, we try to tackle problems by building a model based on training examples. The model can then be used to make predictions or decisions for new examples, avoiding the need to program the desired behavior explicitly. More concretely, given a set of training examples $\mathcal{D}_{\mathcal{T}}$ for task $\mathcal{T}$, we try to obtain a model by optimizing a performance criterion $\mathcal{L}_{\mathcal{T}}$ for the task. In computer vision, the model is typically represented by a convolutional neural network [49, 84, 85] f_θ with learnable parameters θ and trained with the backpropagation algorithm.

The dominant paradigm is to learn one task at a time. For example, we train a separate neural network for detecting objects, estimating depth and predicting attributes. In this case, each neural network is a function of the same input image and produces one output. Since the neural networks are disconnected, it is not possible for tasks to utilize the - possibly relevant - information that is being extracted by other tasks. Figure 2.1 visualizes such a *single-task learning* setup.

Differently, a *multi-task network* [17] has the same inputs as the single-task networks, but infers all desired task outputs together. Figure 2.2 shows a schematic diagram. The multi-task network is trained by applying backpropagation in parallel on all task

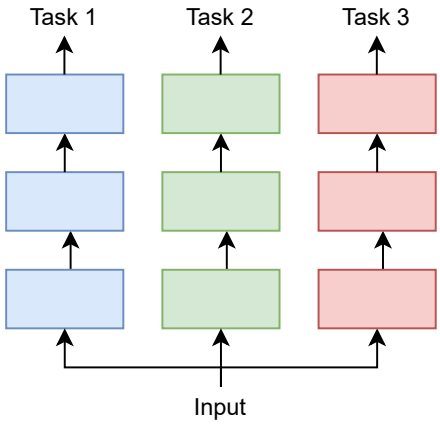


Figure 2.1: Single-task learning of three different tasks with the same inputs. Each of the neural networks is optimized independently.

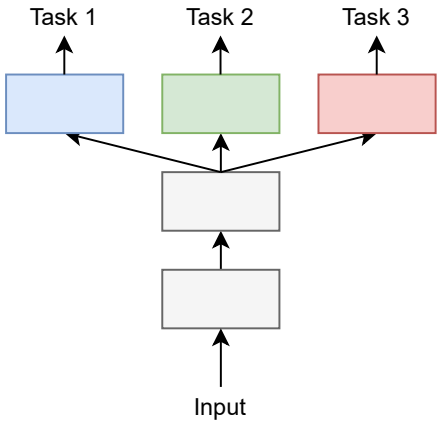


Figure 2.2: Multi-task learning of three different tasks. We optimize all tasks in parallel, while sharing the learned representations.

outputs. The multi-task learning objective is defined as

$$\mathcal{L}_{MTL} = \sum_i w_i \cdot \mathcal{L}_i \tag{2.1}$$

where $\mathcal{L}_i$ denotes the task-specific loss for task i and w_i is a task-specific weight. Since the tasks share a number of layers in the network, the internal representations that are being learned for one task can also be utilized by other tasks. Learning tasks in parallel while sharing the learned representations is the key idea in multi-task learning.

Caruana [17] showed that multi-task learning nets can improve generalization by using information from the training signals of related tasks. In particular, he identified several mechanisms that allow to improve generalization. We provide a brief overview below:

- **Data amplification:** We effectively increase the size of the training data due to the additional information in the training signals of related tasks.
- **Eavesdropping:** Let F be a feature, useful to the tasks T and T', that is easy to pick up when learning task T but difficult to discover when learning T'. In a MTL net, task T' could eavesdrop on the features learned for T, e.g. F, and obtain better performance.
- **Attributes selection:** Assume two tasks T and T' that use a common feature F. In this case, the MTL net will be better at selecting data attributes relevant to F because there is a stronger training signal for these attributes.

- **Representation bias:** Representations that allow to solve multiple tasks better represent the data's regularities.
- **Overfitting:** Tasks are less likely to overfit onto a feature F if we train them together.

We refer to the original work for a detailed analysis.

The challenges for training multi-task learning models are two-fold. First, we need to design appropriate architectures that can handle multiple tasks. Second, we need to come up with effective ways of jointly optimizing multiple objectives.

2.2 Datasets

We perform our experimental evaluation on a variety of benchmarks. The included datasets are of varying size, cover different task dictionaries and contain annotations that were provided by human annotators or obtained through distillation. This diverse group of settings allows us to scrutinize the pros and cons of the methods under consideration. We describe the different benchmarks below.

The **PASCAL** dataset [45] is a popular benchmark for dense prediction tasks. We use the split from PASCAL-Context [26] which has annotations for semantic segmentation, human part segmentation and semantic edge detection. Additionally, we consider the tasks of surface normals prediction and saliency detection. The annotations were distilled by [110] using pre-trained state-of-the-art models [9, 20].

The **Cityscapes** dataset [29] considers the scenario of urban scene understanding. The train, validation and test set contain 2975, 500 and 1525 images respectively, taken by driving a car in European cities. We consider a number of dense prediction tasks on the Cityscapes dataset: semantic segmentation, instance segmentation and monocular depth estimation. The depth maps were generated with SGM [61].

The **NYUD** dataset [140] considers indoor scene understanding. The dataset contains 795 train and 654 test images annotated for semantic segmentation and monocular depth estimation. Other works have also considered surface normal prediction [165, 179, 110] and semantic edge detection [165, 110] on the NYUD-v2 dataset. The annotations for these tasks can be directly derived from the semantic and depth ground truth. In this manuscript we focus on the semantic segmentation and depth estimation tasks.

The **Taskonomy** dataset [176] contains semi-real images of indoor scenes, annotated for 26 (dense prediction, classification, etc.) tasks. Out of all available tasks, we select scene categorization, semantic segmentation, edge detection, monocular depth estimation and keypoint detection for our experiments. The task dictionary was selected

Table 2.1: Overview of the multi-task learning benchmarks. Distilled task labels are marked with an asterisk.

Dataset	#Train	#Val	#Test	Segm.	Depth	Inst.	Parts	Norm.	Sal.	Edges	Keyp.	Scene	Attr.
PASCAL	10,581	1,449	-	✓			✓	✓*	✓*	✓			
Cityscapes	2,975	500	500	✓	✓*	✓							
NYUD	795	654	-	✓	✓			✓		✓			
Taskonomy	275k	52k	54k	✓*	✓					✓*	✓*	✓*	
CelebA	160k	20k	20k										✓

Segm.: Semantic Segmentation, Inst.: Instance Segmentation, Parts: Human Parts Segmentation, Norm: Surface Normals Prediction, Sal.: Saliency Estimation Edges: Semantic Edges or Boundaries, Keyp: 2D-Keypoint Estimation, Scene: Scene Categorization, Attr.: Person Attributes Classification.

to be as diverse as possible, while still keeping the total number of tasks reasonably low for all required computations. We use the tiny split of the dataset, containing 275k train, 52k validation and 54k test images.

The **CelebA** dataset [102] contains over 200k images of celebrities labeled with 40 facial attribute categories. The training, validation and test set contain 160k, 20k and 20k images respectively. While this dataset is mainly suitable for multi-attribute learning, prior works [107, 139, 152] used it for multi-task learning by treating the prediction of each facial attribute as a separate binary classification task. This universal use of a binary cross-entropy loss across all tasks makes CelebA an interesting setting to compare the influence of different strategies for setting the task-specific weights in Equation 2.1, thus we chose to also include it in our experiments.

Table 2.1 shows an overview of the datasets. Figure 2.3 shows a few example images from the different datasets.

2.3 Evaluation Criteria

In addition to reporting the performance on every individual task, we include a single-number performance metric for the multi-task models. We define the multi-task performance Δ_{MTL} of a multi-task learning model m as the average per-task drop or increase in performance w.r.t. the single-task baseline b:

$$\Delta_{MTL} = \frac{1}{T}\sum_{i=1}^{T}(-1)^{l_i}\left(M_{m,i} - M_{b,i}\right)/M_{b,i}, \tag{2.2}$$

where $l_i = 1$ if a lower value means better performance for metric M_i of task i, and 0 otherwise. The single-task performance is measured for a fully-converged model that uses the same backbone network only to perform that task. To achieve a fair comparison, we report the results after performing a search on the most important hyperparameters: optimizer, initial learning rate, learning rate scheduler and weight decay. Hyperparameters that are method-specific are set in accordance with the

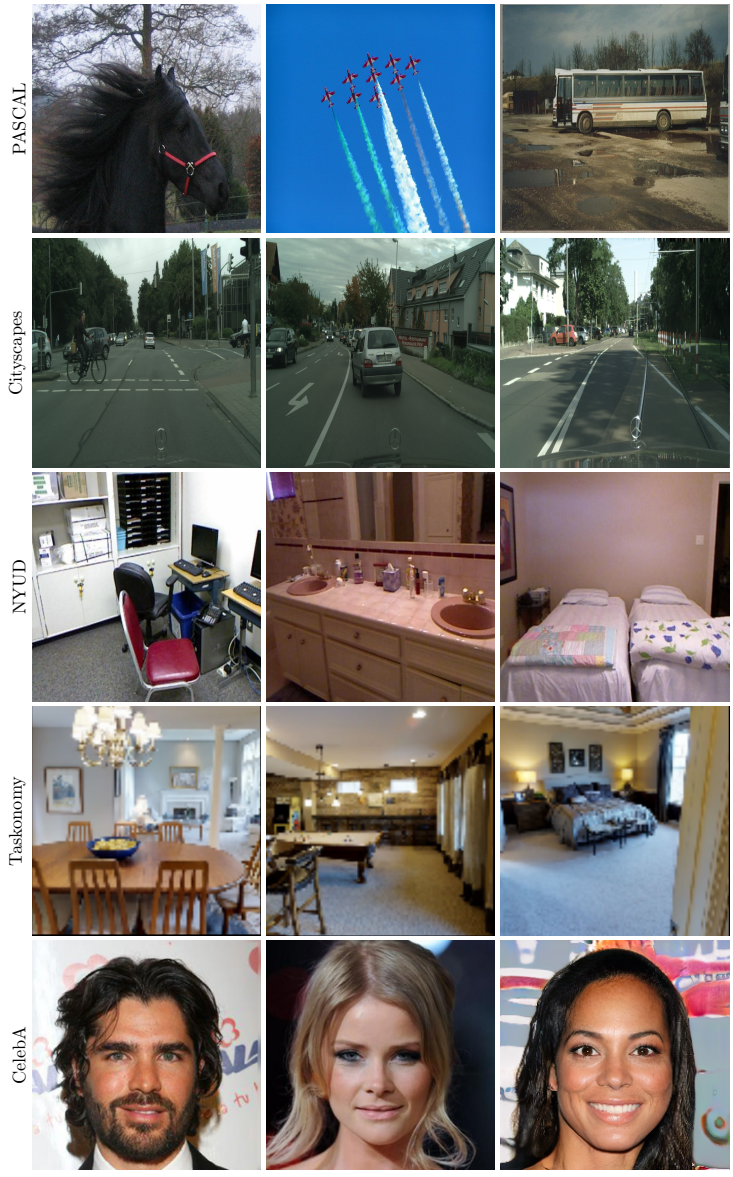


Figure 2.3: A few example images. The aspect ratios can differ from the originals due to the visualization.

setup advised by the authors. This practice ensures that every model is trained with comparable amounts of finetuning.

The MTL performance metric does not account for the variance when different hyperparameters are used. To address this, we analyze the influence of the used hyperparameters with performance profiles for some experiments. Finally, in addition to a performance evaluation, we also include the model resources, i.e. number of parameters and FLOPS, when comparing the multi-task architectures.

2.4 Conclusion

We have described the problem setup, the benchmarks and the evaluation criteria that will be used in this manuscript. The most important challenges for multi-task learning consist of designing appropriate architectures and finding effective training recipes. The next chapter summarizes prior works that considered these problems.

Chapter 3

Related Work

The objective of this chapter is two-fold. First, we aim to provide a structured overview of state-of-the-art MTL techniques for computer vision. Second, starting from the proposed taxonomy, we try to motivate the different lenses through which we study the multi-task learning problem in this thesis.

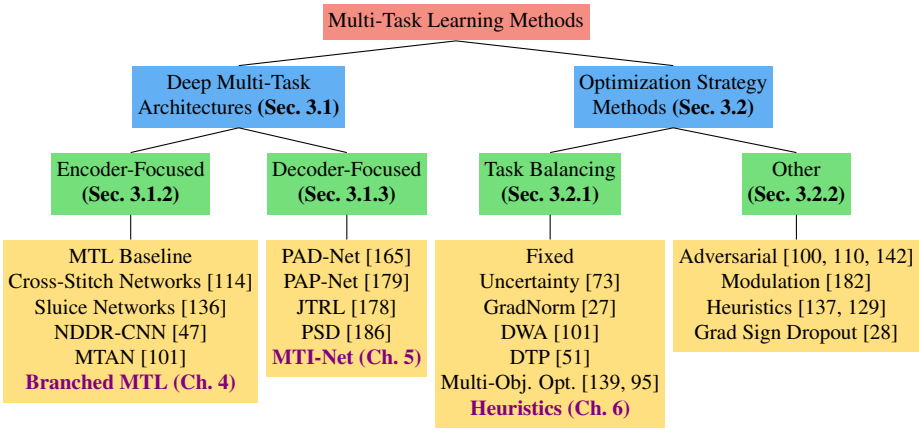
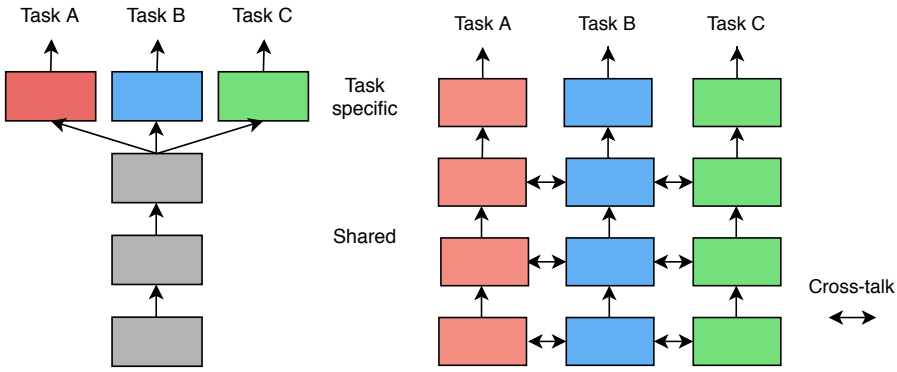


Figure 3.1: A taxonomy of recent deep learning approaches for jointly solving multiple visual recognition tasks. We indicate **methods** that are proposed in this manuscript.

Figure 3.1 shows an overview of the methods that we will review. Section 3.1 considers different deep multi-task architectures, categorizing them into two main groups: encoder- and decoder-focused approaches. Section 3.2 surveys various optimization techniques for balancing the influence of the tasks when updating the network's weights.



(a) Hard parameter sharing (b) Soft parameter sharing

Figure 3.2: Historically multi-task learning using deep neural networks has been subdivided into soft- and hard-parameter sharing schemes.

We consider the majority of task balancing, adversarial and modulation techniques. Section 3.3 concludes our literature review.

The content of this chapter is based upon the following publication:

- *Vandenhende, S., Georgoulis, S., Van Gansbeke, W., Proesmans, M., Dai, D., & Van Gool, L. (2021). Multi-task learning for dense prediction tasks: A survey. IEEE Transactions on Pattern Analysis and Machine Intelligence.*

3.1 Deep Multi-Task Architectures

In this section, we review deep multi-task architectures used in computer vision. First, we give a brief historical overview of MTL approaches, before introducing a novel taxonomy to categorize different methods. Second, we discuss network designs from different groups of works, and analyze their advantages and disadvantages. Note that, as a detailed presentation of each architecture is beyond the scope of this work, in each case we refer the reader to the corresponding paper for further details that complement the following descriptions.

3.1.1 Historical Overview and Taxonomy

Non-Deep Learning Methods

Before the deep learning era, MTL works tried to model the common information among tasks in the hope that a joint task learning would result in better generalization performance. To achieve this, they placed assumptions on the task parameter space, such as: task parameters should lie close to each other w.r.t. some distance metric [46, 167, 69, 185], share a common probabilistic prior [8, 173, 86, 32, 81], or reside in a low dimensional subspace [5, 99, 70] or manifold [2]. These assumptions work well when all tasks are related [46, 4, 5, 130], but can lead to performance degradation if information sharing happens between unrelated tasks. The latter is a known problem in MTL, referred to as *negative transfer*. To mitigate this problem, some of these works opted to cluster tasks into groups based on prior beliefs about their similarity or relatedness.

Soft and Hard Parameter Sharing in Deep Learning

In the context of deep learning, MTL is performed by learning shared representations from multi-task supervisory signals. Historically, deep multi-task architectures were classified into hard or soft parameter sharing techniques. In *hard parameter sharing*, the parameter set is divided into shared and task-specific parameters (see Figure 3.2a). MTL models using hard parameter sharing typically consist of a shared encoder that branches out into task-specific heads [117, 73, 27, 139, 145]. In *soft parameter sharing*, each task is assigned its own set of parameters and a feature sharing mechanism handles the cross-task talk (see Figure 3.2b). We summarize representative works for both groups of works below.

Hard Parameter Sharing UberNet [77] was the first hard-parameter sharing model to jointly tackle a large number of low-, mid-, and high-level vision tasks. The model featured a multi-head design across different network layers and scales. Still, the most characteristic hard parameter sharing design consists of a shared encoder that branches out into task-specific decoding heads [117, 73, 27, 139, 145]. Multilinear relationship networks [105] extended this design by placing tensor normal priors on the parameter set of the fully connected layers. In a different vein, stochastic filter grouping [13] re-purposes the convolution kernels in each layer to support shared or task-specific behavior.

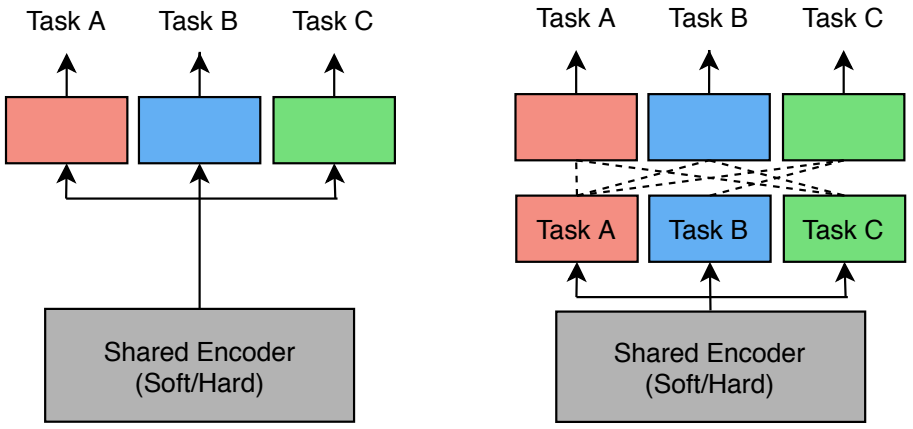
Soft Parameter Sharing Cross-stitch networks [114] introduced soft-parameter sharing in deep MTL architectures. The model uses a linear combination of the

activations in every layer of the task-specific networks as a means for soft feature fusion. Sluice networks [136] extended this idea by allowing to learn the selective sharing of layers, subspaces and skip connections. NDDR-CNN [47] also incorporated dimensionality reduction techniques into the feature fusion layers. Differently, MTAN [101] used an attention mechanism to share a general feature pool amongst the task-specific networks. A concern with soft parameter sharing approaches is scalability, as the size of the multi-task network tends to grow linearly with the number of tasks.

Distilling Task Predictions in Deep Learning

All deep learning works presented so far follow a common pattern: they *directly* predict all task outputs from the same input in one processing cycle. In this case, we rely entirely on the backpropagation algorithm to find the right representation to solve the tasks. However, it's unlikely that such an approach will be able to discover all useful relationships between tasks. For example, the model might fail to align discontinuities for tasks like depth estimation and semantic segmentation. For this reason, a few recent works first employed a multi-task network to make initial task predictions, and then leveraged features from these initial predictions to further improve each task output – in a one-off or recursive manner. In this case, the task relationships are modeled in a more explicit way, which makes it easier to exploit correlations between tasks that allow to improve the performance.

Before the deep learning era, Hoiem *et al.* [62] applied the idea of recursively predicting tasks to develop an integrated 3D scene understanding system with estimates of surface orientations, occlusion boundaries, objects, camera viewpoint, and relative depth. Later, Multi-Net [11] re-introduced the concept in the context of deep neural networks. Opposed to encoder-focused approaches, Multi-Net does not only share the features between tasks, but also allows tasks to interact in a recursive manner by encoding the task predictions in a common shared representation. PAD-Net [165] proposed to distill information from the initial task predictions of other tasks, by means of spatial attention, before adding it as a residual to the task of interest. JTRL [178] opted for sequentially predicting each task, with the intention to utilize information from the past predictions of one task to refine the features of another task at each iteration. PAP-Net [179] extended upon this idea, and used a recursive procedure to propagate similar cross-task and task-specific patterns found in the initial task predictions. To do so, they operated on the affinity matrices of the initial predictions, and not on the features themselves, as was the case before [165, 178]. Zhou et al. [186] refined the use of pixel affinities to distill the information by separating inter- and intra-task patterns from each other.



(a) Encoder-focused model

(b) Decoder-focused model

Figure 3.3: In this work we discriminate between encoder- and decoder-focused models depending on where the task interactions take place.

A New Taxonomy of MTL Approaches

As explained before, multi-task networks have historically been classified into soft or hard parameter sharing techniques. However, several recent works took inspiration from both groups of works to jointly solve multiple tasks. As a consequence, it is debatable whether the soft versus hard parameter sharing paradigm should still be used as the main framework for classifying MTL architectures. In this work, we propose an alternative taxonomy that discriminates between different architectures on the basis of where the *task interactions* take place, i.e. locations in the network where information or features are exchanged or shared between tasks. The impetus for this framework was given in the previous subsection. Based on the proposed criterion, we distinguish between two types of models: *encoder-focused* and *decoder-focused* architectures. The encoder-focused architectures (see Figure 3.3a) only share information in the encoder, using either hard- or soft-parameter sharing, before decoding each task with an independent task-specific head. Differently, the decoder-focused architectures (see Figure 3.3b) also exchange information during the decoding stage. In particular, they first make initial task predictions, and then leverage features from these initial task predictions to improve each task output - in a sequential manner. Figure 3.1 gives an overview of the proposed taxonomy, listing representative works in each case.

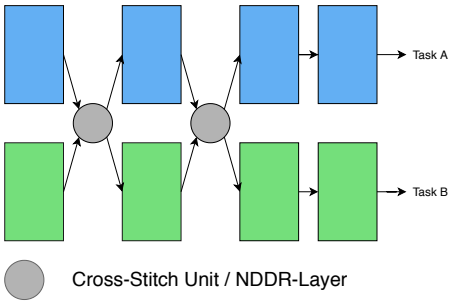


Figure 3.4: The architecture of cross-stitch networks [114] and NDDR-CNNs [47]. The activations from all single-task networks are fused across several encoding layers. Different feature fusion mechanisms were used in each case.

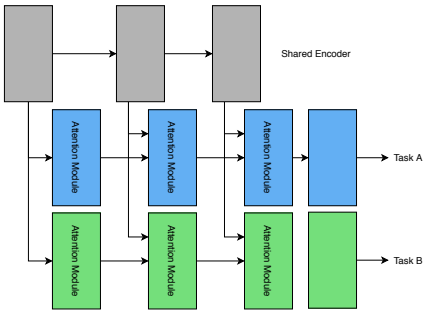


Figure 3.5: The architecture of MTAN [101]. Task-specific attention modules select and refine features from several layers of a shared encoder.

3.1.2 Encoder-focused Architectures

Encoder-focused architectures (see Figure 3.3a) share the task features in the encoding stage, before they process them with a set of independent task-specific heads. A number of works [117, 73, 27, 139, 145] followed an ad hoc strategy by sharing an off-the-shelf backbone network in combination with small task-specific heads (see Figure 3.2a). This model relies on the encoder (i.e. backbone network) to learn a generic representation of the scene. The features from the encoder are then used by the task-specific heads to get the predictions for every task. While this simple model shares the full encoder amongst all tasks, recent works have considered *where* and *how* the feature sharing should happen in the encoder. We discuss such sharing strategies in the following sections.

Cross-Stitch Networks

Cross-stitch networks [114] shared the activations amongst all single-task networks in the encoder. Assume we are given two activation maps x_A, x_B at a particular layer, that belong to tasks A and B respectively. A learnable linear combination of these activation maps is applied, before feeding the transformed result $\tilde{x}_A, \tilde{x}_B$ to the next layer in the single-task networks. The transformation is parameterized by learnable scalars $\alpha \in \mathbb{R}$, and can be expressed as

$$\begin{bmatrix} \tilde{x}_A \\ \tilde{x}_B \end{bmatrix} = \begin{bmatrix} \alpha_{AA} & \alpha_{AB} \\ \alpha_{BA} & \alpha_{BB} \end{bmatrix} \begin{bmatrix} x_A \\ x_B \end{bmatrix}. \tag{3.1}$$

As illustrated in Figure 3.4, this procedure is repeated at multiple locations in the encoder. By learning the weights α, the network can decide the degree to which the features are shared between tasks. In practice, we are required to pre-train the single-task networks, before stitching them together, in order to maximize the performance. A disadvantage of cross-stitch networks is that the size of the network increases linearly with the number of tasks. Furthermore, it is not clear where the cross-stitch units should be inserted in order to maximize their effectiveness. **Sluice networks** [136] extended this work by also supporting the selective sharing of subspaces and skip connections.

Neural Discriminative Dimensionality Reduction

Neural Discriminative Dimensionality Reduction CNNs (NDDR-CNNs) [47] used a similar architecture with cross-stitch networks (see Figure 3.4). However, instead of utilizing a linear combination to fuse the activations from all single-task networks, a dimensionality reduction mechanism is employed. First, features with the same spatial resolution in the single-task networks are concatenated channel-wise. Second, the number of channels is reduced by processing the features with a 1 by 1 convolutional layer, before feeding the result to the next layer. The convolutional layer allows to fuse activations across all channels. Differently, cross-stitch networks only allow to fuse activations from channels that share the same index. The NDDR-CNN behaves as a cross-stitch network when the non-diagonal elements in the weight matrix of the convolutional layer are zero.

Due to their similarity with cross-stitch networks, NDDR-CNNs are prone to the same problems. First, there is a scalability concern when dealing with a large number of tasks. Second, NDDR-CNNs involve additional design choices, since we need to decide where to include the NDDR layers. Finally, both cross-stitch networks and NDDR-CNNs only allow to use limited local information (i.e. small receptive field) when fusing the activations from the different single-task networks. We hypothesize that this is suboptimal because the use of sufficient context is very important during encoding – as already shown for the tasks of image classification [58] and semantic segmentation [19, 180, 174]. This is backed up by certain decoder-focused architectures in Section 3.1.3 that overcome the limited receptive field by predicting the tasks at multiple scales and by sharing the features repeatedly at every scale.

Multi-Task Attention Networks

Multi-Task Attention Networks (MTAN) [101] used a shared backbone network in conjunction with task-specific attention modules in the encoder (see Figure 3.5). The shared backbone extracts a general pool of features. Then, each task-specific attention module selects features from the general pool by applying a soft attention mask. The attention mechanism is implemented using regular convolutional layers and a sigmoid non-linearity. Since the attention modules are small compared to the backbone network, the MTAN model does not suffer as severely from the scalability issues that are typically associated with cross-stitch networks and NDDR-CNNs. However, similar to the fusion mechanism in the latter works, the MTAN model can only use limited local information to produce the attention mask.

Towards Branched Multi-Task Networks (Chapter 4)

Most of the encoder-focused architectures softly share the features amongst tasks during the encoding stage. These methods report good performance, but require a considerable amount of computational resources. In contrast, hard parameter sharing models consume fewer resources, but report lower performance compared to their soft parameter sharing equivalents. The latter can be explained as follows. Assuming a hard parameter sharing setting, the number of possible network configurations grows quickly with the number of tasks. Therefore, previous works on hard parameter sharing opted for the simple strategy of sharing the initial layers in the network, after which all tasks branch out simultaneously. The point at which the branching occurs is usually determined ad hoc [73, 51, 139]. This situation hurts performance, as a suboptimal grouping of tasks will lead to the sharing of information between unrelated tasks.

Branched Multi-Task Networks present an approach to decide on the degree of layer sharing between tasks in order to eliminate the need for manual exploration. To this end, we base the layer sharing on measurable levels of *task affinity* or *task relatedness*: two tasks are strongly related, if their single task models rely on a similar set of features. Additionally, our method allows to trade network complexity against task similarity. We provide extensive empirical evaluation of our method, showing its superiority in terms of multi-task performance versus computational resources.

3.1.3 Decoder-Focused Architectures

The encoder-focused architectures in Section 3.1.2 follow a common pattern: they *directly* predict all task outputs from the same input in one processing cycle (i.e. all predictions are generated once, in parallel or sequentially, and are not refined afterwards). By doing so, they fail to capture commonalities and differences among

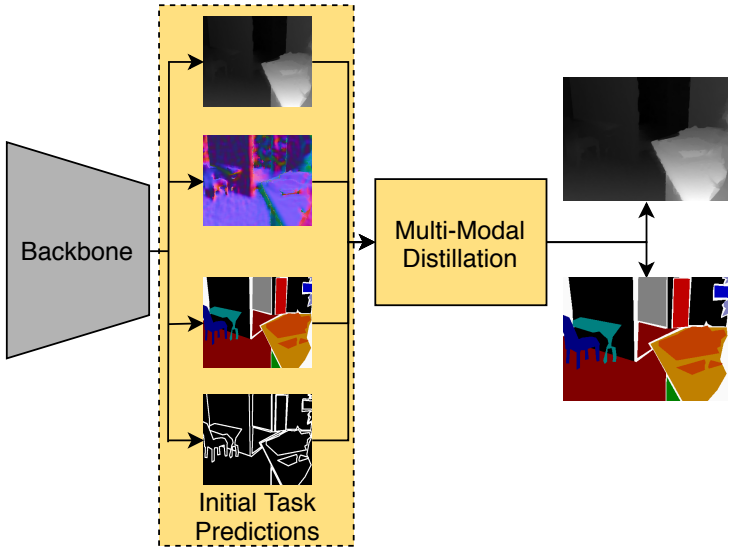


Figure 3.6: The architecture in PAD-Net [165]. Features extracted by a backbone network are passed to task-specific heads to make initial task predictions. The task features from the different heads are then combined through a distillation unit to make the final predictions. Note that, auxiliary tasks can be used in this framework, i.e. tasks for which only the initial predictions are generated, but not the final ones.

tasks, that are likely fruitful for one another (e.g. depth discontinuities are usually aligned with semantic edges). Arguably, this might be the reason for the moderate only performance improvements achieved by the encoder-focused approaches to MTL. To alleviate this issue, a few recent works first employed a multi-task network to make initial task predictions, and then leveraged features from these initial predictions in order to further improve each task output – in an one-off or recursive manner. As these MTL approaches also share or exchange information during the decoding stage, we refer to them as decoder-focused architectures (see Figure 3.3b).

PAD-Net

PAD-Net [165] was one of the first decoder-focused architectures. The model itself is visualized in Figure 3.6. As can be seen, the input image is first processed by an off-the-shelf backbone network. The backbone features are further processed by a set of task-specific heads that produce an initial prediction for every task. These initial task predictions add deep supervision to the network, but they can also be used to exchange information between tasks, as will be explained next. The *task features* in the last

layer of the task-specific heads contain a per-task feature representation of the scene. PAD-Net proposed to re-combine them via a multi-modal distillation unit, whose role is to extract cross-task information, before producing the final task predictions.

PAD-Net performs the multi-modal distillation by means of a spatial attention mechanism. Particularly, the output features F_k^o for task k are calculated as

$$F_k^o = F_k^i + \sum_{l \neq k} \sigma\left(W_{k,l} F_l^i\right) \odot F_l^i, \tag{3.2}$$

where $\sigma\left(W_{k,l} F_l^i\right)$ returns a spatial attention mask that is applied to the initial task features F_l^i from task l. The attention mask itself is found by applying a convolutional layer $W_{k,l}$ to extract local information from the initial task features. Equation 3.2 assumes that the task interactions are location dependent, i.e. tasks are not in a constant relationship across the entire image. This can be understood from a simple example. Consider two dense prediction tasks, e.g. monocular depth prediction and semantic segmentation. Depth discontinuities and semantic boundaries often coincide. However, when we segment a flat object, e.g. a magazine, from a flat surface, e.g. a table, we will still find a semantic boundary where the depth map is rather continuous. In this particular case, the depth features provide no additional information for the localization of the semantic boundaries. The use of spatial attention explicitly allows the network to select information from other tasks at locations where its useful.

The encoder-focused approaches in Section 3.1.2 shared features amongst tasks using the intermediate representations in the encoder. Differently, PAD-Net models the task interactions by applying a spatial attention layer to the features in the task-specific heads. In contrast to the intermediate feature representations in the encoder, the task features used by PAD-Net are already disentangled according to the output task. We hypothesize that this makes it easier for other tasks to distill the relevant information. This multi-step decoding strategy from PAD-Net is applied and refined in other decoder-focused approaches.

Pattern-Affinitive Propagation Networks

Pattern-Affinitive Propagation Networks (PAP-Net) [179] used an architecture similar to PAD-Net (see Figure 3.7), but the multi-modal distillation in this work is performed in a different manner. The authors argue that directly working on the task features space via the spatial attention mechanism, as done in PAD-Net, might be a suboptimal choice. As the optimization still happens at a different space, i.e. the task label space, there is no guarantee that the model will learn the desired task relationships. Instead, they statistically observed that pixel affinities tend to align well with common local structures on the task label space. Motivated by this observation, they proposed to leverage pixel affinities in order to perform multi-modal distillation.

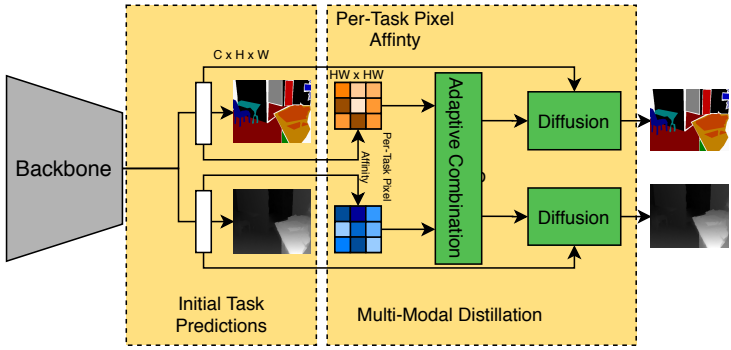


Figure 3.7: The architecture in PAP-Net [179]. Features extracted by a backbone network are passed to task-specific heads to make initial task predictions. The task features from the different heads are used to calculate a per-task pixel affinity matrix. The affinity matrices are adaptively combined and diffused back into the task features space to spread the cross-task correlation information across the image. The refined features are used to make the final task predictions.

To achieve this, the backbone features are first processed by a set of task-specific heads to get an initial prediction for every task. Second, a per-task pixel affinity matrix M_{T_j} is calculated by estimating pixel-wise correlations upon the task features coming from each head. Third, a cross-task information matrix $\hat{M}_{T_j}$ for every task T_j is learned by adaptively combining the affinity matrices M_{T_i} for tasks T_i with learnable weights $\alpha_i^{T_j}$

$$\hat{M}_{T_j} = \sum_{T_i} \alpha_i^{T_j} \cdot M_{T_i}. \tag{3.3}$$

Finally, the task features coming from each head j are refined using the cross-task information matrix $\hat{M}_{T_j}$. In particular, the cross-task information matrix is diffused into the task features space to spread the correlation information across the image. This effectively weakens or strengthens the pixel correlations for task T_j, based on the pixel affinities from other tasks T_i. The refined features are used to make the final predictions for every task.

Joint Task-Recursive Learning

Joint Task-Recursive Learning (JTRL) [178] recursively predicts two tasks at increasingly higher scales in order to gradually refine the results based on past states. The architecture is illustrated in Figure 3.8. Similarly to PAD-Net and PAP-Net, a multi-modal distillation mechanism is used to combine information from earlier task

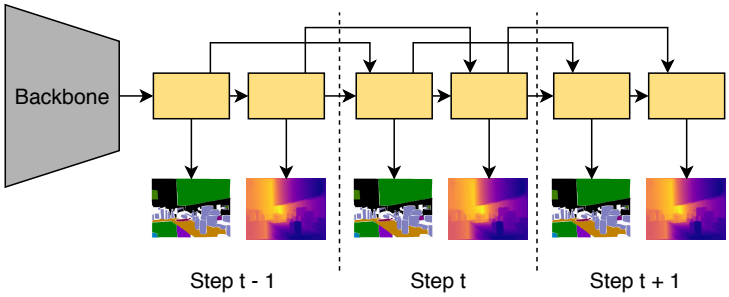


Figure 3.8: The architecture in Joint Task-Recursive Learning [178]. The features of two tasks are progressively refined in an intertwined manner based on past states.

predictions, through which later predictions are refined. Differently, the JTRL model predicts two tasks sequentially, rather than in parallel, and in an intertwined manner. The main disadvantage of this approach is that it is not straightforward, or even possible, to extent this model to more than two tasks given the intertwined manner at which task predictions are refined.

Towards Multi-Scale Task Interaction Networks (Chapter 5)

The decoder-focused architectures perform multi-modal distillation at a fixed scale, e.g. the features of the backbone's last layer. This rests on the assumption that all relevant task interactions can solely be modeled through a single filter operation with a specific receptive field. In this manuscript, we show that this is a rather strict assumption. In fact, tasks can influence each other differently at different receptive fields. To account for this, we propose a **Multi-Scale Task Interaction Network (MTI-Net)** that explicitly takes into account task interactions at multiple scales. This research is presented in Chapter 5.

3.1.4 Other Approaches

A number of approaches that fall outside the aforementioned categories have been proposed in the literature. For example, multilinear relationship networks [105] used tensor normal priors for the parameter set of the task-specific heads to allow interactions in the decoding stage. Different from the standard parallel ordering scheme, where layers are aligned and shared (e.g. [114, 47]), soft layer ordering [113] proposed a flexible sharing scheme across tasks and network depths. Yang et al. [169] generalized matrix factorisation approaches to MTL in order to learn cross-task sharing structures in every layer of the network. Routing networks [133] proposed a principled

approach to determine the connectivity of a network's function blocks through routing. Piggyback [108] showed how to adapt a single, fixed neural network to a multi-task network by learning binary masks. Huang et al. [66] introduced a method rooted in Neural Architecture Search (NAS) for the automated construction of a tree-based multi-attribute learning network. Stochastic filter grouping [13] re-purposed the convolution kernels in each layer of the network to support shared or task-specific behaviour. In a similar vein, feature partitioning [118] presented partitioning strategies to assign the convolution kernels in each layer of the network into different tasks. Attentive Single-Tasking of Multiple Tasks (ASTMT) [110] proposed to take a 'single-tasking' route for the MTL problem. That is, within a multi-tasking framework they perform separate forward passes, one for each task, that activate shared responses among all tasks, plus some residual responses that are task-specific.

3.2 Optimization in MTL

In the previous section, we discussed the construction of network architectures that are able to learn multiple tasks concurrently. Still, a significant challenge in MTL stems from the optimization procedure itself. In particular, we need to carefully balance the joint learning of all tasks to avoid a scenario where one or more tasks have a dominant influence in the network weights. In this section, we discuss several methods that have considered this *task balancing* problem.

3.2.1 Task Balancing Approaches

Without loss of generality, the optimization objective in a MTL problem, assuming task-specific weights w_i and task-specific loss functions $\mathcal{L}_i$, can be formulated as

$$\mathcal{L}_{MTL} = \sum_i w_i \cdot \mathcal{L}_i. \tag{3.4}$$

When using stochastic gradient descent to minimize the objective from Equation 3.4, which is the standard approach in the deep learning era, the network weights in the shared layers W_{sh} are updated by the following rule

$$W_{sh} = W_{sh} - \gamma \sum_i w_i \frac{\partial \mathcal{L}_i}{\partial W_{sh}}. \tag{3.5}$$

From Equation 3.5 we can draw the following conclusions. First, the network weight update can be suboptimal when the task gradients conflict, or dominated by one task when its gradient magnitude is much higher w.r.t. the other tasks. This motivated

researchers [73, 51, 101, 27] to balance the gradient magnitudes by setting the task-specific weights w_i in the loss. To this end, other works [139, 182, 144] have also considered the influence of the direction of the task gradients. Second, each task's influence on the network weight update can be controlled, either *indirectly* by adapting the task-specific weights w_i in the loss, or *directly* by operating on the task-specific gradients $\frac{\partial \mathcal{L}_i}{\partial W_{sh}}$. A number of methods that tried to address these problems are discussed next.

Uncertainty Weighting

Kendall et al. [73] used the **homoscedastic uncertainty** to balance the single-task losses. The homoscedastic uncertainty or *task-dependent uncertainty* is not an output of the model, but a quantity that remains constant for different input examples of the same task. The optimization procedure is carried out to maximize a Gaussian likelihood objective that accounts for the homoscedastic uncertainty. In particular, they optimize the model weights W and the noise parameters σ_1, σ_2 to minimize the following objective

$$\mathcal{L}\left(W, \sigma_1, \sigma_2\right) = \frac{1}{2\sigma_1^2}\mathcal{L}_1\left(W\right) + \frac{1}{2\sigma_2^2}\mathcal{L}_2\left(W\right) + \log \sigma_1 \sigma_2. \tag{3.6}$$

The loss functions $\mathcal{L}_1$, $\mathcal{L}_2$ belong to the first and second task respectively. By minimizing the loss $\mathcal{L}$ w.r.t. the noise parameters σ_1, σ_2, one can essentially balance the task-specific losses during training. The optimization objective in Equation 3.6 can easily be extended to account for more than two tasks too. The noise parameters are updated through standard backpropagation during training.

Note that, increasing the noise parameter σ_i reduces the weight for task i. Consequently, the effect of task i on the network weight update is smaller when the task's homoscedastic uncertainty is high. This is advantageous when dealing with noisy annotations since the task-specific weights will be lowered automatically for such tasks.

Gradient Normalization

Gradient normalization (GradNorm) [27] proposed to control the training of multi-task networks by stimulating the task-specific gradients to be of similar magnitude. By doing so, the network is encouraged to learn all tasks at an equal pace. Before presenting this approach, we introduce the necessary notations in the following paragraph.

We define the L2 norm of the gradient for the weighted single-task loss $w_i\left(t\right) \cdot L_i\left(t\right)$ at step t w.r.t. the weights W, as $G_i^W\left(t\right)$. We additionally define the following quantities,

- the mean task gradient $\bar{G}^W$ averaged across all task gradients G_i^W w.r.t the weights W at step t: $\bar{G}^W(t) = E_{task}\left[G_i^W(t)\right]$;
- the inverse training rate $\tilde{L}_i$ of task i at step t: $\tilde{L}_i(t) = L_i(t)/L_i(0)$;
- the relative inverse training rate of task i at step t: $r_i(t) = \tilde{L}_i(t)/E_{task}\left[\tilde{L}_i(t)\right]$.

GradNorm aims to balance two properties during the training of a multi-task network. First, balancing the gradient magnitudes G_i^W. To achieve this, the mean gradient $\bar{G}^W$ is considered as a common basis from which the relative gradient sizes across tasks can be measured. Second, balancing the pace at which different tasks are learned. The relative inverse training rate $r_i(t)$ is used to this end. When the relative inverse training rate $r_i(t)$ increases, the gradient magnitude $G_i^W(t)$ for task i should increase as well to stimulate the task to train more quickly. GradNorm tackles both objectives by minimizing the following loss

$$\left|G_i^W(t) - \bar{G}^W(t) \cdot r_i(t)\right|. \tag{3.7}$$

Remember that, the gradient magnitude $G_i^W(t)$ for task i depends on the weighted single-task loss $w_i(t) \cdot L_i(t)$. As a result, the objective in Equation 3.7 can be minimized by adjusting the task-specific weights w_i. In practice, during training these task-specific weights are updated in every iteration using backpropagation. After every update, the task-specific weights $w_i(t)$ are re-normalized in order to decouple the learning rate from the task-specific weights.

Note that, calculating the gradient magnitude $G_i^W(t)$ requires a backward pass through the task-specific layers of every task i. However, savings on computation time can be achieved by considering the task gradient magnitudes only w.r.t. the weights in the last shared layer.

Different from uncertainty weighting, GradNorm does not take into account the task-dependent uncertainty to re-weight the task-specific losses. Rather, GradNorm tries to balance the pace at which tasks are learned, while avoiding gradients of different magnitude.

Dynamic Weight Averaging

Similar to GradNorm, Liu et al. [101] proposed a technique, termed **Dynamic Weight Averaging** (DWA), to balance the pace at which tasks are learned. Differently, DWA only requires access to the task-specific loss values. This avoids having to perform separate backward passes during training in order to obtain the task-specific gradients.

In DWA, the task-specific weight w_i for task i at step t is set as

$$w_i(t) = \frac{N \exp\left(r_i(t-1)/T\right)}{\sum_n \exp\left(r_n(t-1)/T\right)}, r_n(t-1) = \frac{L_n(t-1)}{L_n(t-2)}, \tag{3.8}$$

with N being the number of tasks. The scalars $r_n(\cdot)$ estimate the relative descending rate of the task-specific loss values L_n. The temperature T controls the softness of the task weighting in the softmax operator. When the loss of a task decreases at a slower rate compared to other tasks, the task-specific weight in the loss is increased.

Note that, the task-specific weights w_i are solely based on the rate at which the task-specific losses change. Such a strategy requires to balance the overall loss magnitudes beforehand, else some tasks could still overwhelm the others during training. GradNorm avoids this problem by balancing both the training rates and the gradient magnitudes through a single objective (see Equation 3.7).

Dynamic Task Prioritization

The presented task balancing techniques optimized the task-specific weights w_i as part of a Gaussian likelihood objective [73], or in order to balance the pace at which the different tasks are learned [101, 27]. In contrast, **Dynamic Task Prioritization** (DTP) [51] opted to prioritize the learning of 'difficult' tasks by assigning them a higher task-specific weight. The motivation is that the network should spend more effort to learn the 'difficult' tasks. Note that, this is opposed to uncertainty weighting, where a higher weight is assigned to the 'easy' tasks. We hypothesize that the two techniques do not necessarily conflict, but uncertainty weighting seems better suited when tasks have noisy labeled data, while DTP makes more sense when we have access to clean ground-truth annotations.

To measure the task difficulty, one could consider the progress on every task using the loss ratio $\tilde{L}_i(t)$ defined by GradNorm. However, since the loss ratio depends on the initial loss $L_i(0)$, its value can be rather noisy and initialization dependent. Furthermore, measuring the task progress using the loss ratio might not accurately reflect the progress on a task in terms of qualitative results. Therefore, DTP proposes the use of key performance indicators (KPIs) to quantify the difficulty of every task. In particular, a KPI κ_i is selected for every task i, with $0 < \kappa_i < 1$. The KPIs are picked to have an intuitive meaning, e.g. accuracy for classification tasks. For regression tasks, the prediction error can be thresholded to obtain a KPI that lies between 0 and 1. Further, we define a task-level focusing parameter $\gamma_i \geq 0$ that allows to adjust the weight at which easy or hard tasks are down-weighted. DTP sets the task-specific weight w_i for task i at step t as

$$w_i(t) = -\left(1 - \kappa_i(t)\right)^{\gamma_i} \log \kappa_i(t). \tag{3.9}$$

Note that, Equation 3.9 employs a focal loss expression [93] to down-weight the task-specific weights for the 'easy' tasks. In particular, as the value for the KPI κ_i increases, the weight w_i for task i is being reduced.

DTP requires to carefully select the KPIs. For example, consider choosing a threshold to measure the performance on a regression task. Depending on the threshold's value, the task-specific weight will be higher or lower during training. We conclude that the choice of the KPIs in DTP is not determined in a straightforward manner. Furthermore, similar to DWA, DTP requires to balance the overall magnitude of the loss values beforehand. After all, Equation 3.9 does not take into account the loss magnitudes to calculate the task-specific weights. As a result, DTP still involves manual tuning to set the task-specific weights.

MTL as Multi-Objective Optimization

A global optimum for the multi-task optimization objective in Equation 3.4 is hard to find. Due to the complex nature of this problem, a certain choice that improves the performance for one task could lead to performance degradation for another task. The task balancing methods discussed beforehand try to tackle this problem by setting the task-specific weights in the loss according to some heuristic. Differently, Sener and Koltun [139] view MTL as a multi-objective optimization problem, with the overall goal of finding a Pareto optimal solution among all tasks.

In MTL, a Pareto optimal solution is found when the following condition is satisfied: the loss for any task can not be decreased without increasing the loss on any of the other tasks. A **multiple gradient descent algorithm** (MGDA) [37] was proposed in [139] to find a Pareto stationary point. In particular, the shared network weights are updated by finding a common direction among the task-specific gradients. As long as there is a common direction along which the task-specific losses can be decreased, we have not reached a Pareto optimal point yet. An advantage of this approach is that since the shared network weights are only updated along common directions of the task-specific gradients, conflicting gradients are avoided in the weight update step.

Lin et al. [95] observed that MGDA only finds one out of many Pareto optimal solutions. Moreover, it is not guaranteed that the obtained solution will satisfy the users' needs. To address this problem, they generalized MGDA to generate a set of well-representative Pareto solutions from which a preferred solution can be selected. So far, however, the method was only applied to small-scale datasets (e.g. Multi-MNIST).

Heuristics for Balancing Tasks in MTL (Chapter 6)

In Chapter 6, we uncover several discrepancies between existing task balancing strategies. For example, uncertainty weighting sets a lower weight for difficult tasks, while DTP advocates the opposite. We hypothesize that the performance of different task balancing methods is coupled with specific properties of the training dataset. This hypothesis is verified through an extensive experimental comparison of several task balancing approaches. Further, based upon the results, we propose a set of heuristics to set the task-specific weights. The heuristics are of practical use and show more robust results across different benchmarks.

3.2.2 Other Approaches

The task balancing works in Section 3.2.1 can be plugged into most existing multi-task architectures to regulate the task learning. Another group of works also tried to regulate the training of multi-task networks, albeit for more specific setups. We touch upon several of these approaches here. Note that, some of these concepts can be combined with task balancing strategies too.

Zhao et al. [182] empirically found that tasks with gradients pointing in the opposite direction can cause the destructive interference of the gradient. This observation is related to the update rule in Equation 3.5. They proposed to add a modulation unit to the network in order to alleviate the competing gradients issue during training.

Liu et al. [100] considered a specific multi-task architecture where the feature space is split into a shared and a task-specific part. They argue that the shared features should contain more common information, and no information that is specific to a particular task only. The network weights are regularized by enforcing this prior. More specifically, an adversarial approach is used to avoid task-specific features from creeping into the shared representation. Similarly, [142, 110] added an adversarial loss to the single-task gradients in order to make them statistically indistinguishable from each other in the shared parts of the network.

Chen et al. [28] proposed gradient sign dropout, a modular layer that can be plugged into any network with multiple gradient signals. Following MGDA, the authors argue that conflicts in the weight update arise when the gradient values of the different learning signals have opposite signs. Gradient sign dropout operates by choosing the sign of the gradient based on the distribution of the gradient values, and masking out the gradient values with opposite sign. It is shown that the method has several desirable properties, and increases performance and robustness compared to competing works.

Finally, some works relied on heuristics to balance the tasks. Sanh et al. [137] trained the network by randomly sampling a single task for the weight update during every

iteration. The sampling probabilities were set proportionally to the available amount of training data for every task. Raffel et al. [129] used temperature scaling to balance the tasks. So far, however, both procedures were used in the context of natural language processing.

3.3 Conclusion

We reviewed recent methods for multi-task learning within the scope of deep neural networks, including both architectural and optimization based strategies. For each method, we described its key aspects, discussed the commonalities and differences with related works, and presented the possible advantages or disadvantages. Further, starting from the literature, we motivated the various innovations that will be presented in this manuscript:

- Chapter 4 studies Branched Multi-Task Networks - a hard-parameter sharing architecture that achieves a better trade-off between the performance versus computational resources. The proposed method automatically derives a scheme for sharing the layers amongst tasks.
- Chapter 5 shows how decoder-focused architectures can be improved by distilling the task interactions at a multitude of scales. The novel architecture improves over a set of single-tasking baselines in terms of the memory footprint, number of operations and performance.
- Chapter 6 performs an experimental comparison of several task balancing strategies. We unravel what properties of the data render some methods more successful than others. Additionally, we propose a set of heuristics to set the task-specific weights under the multi-task learning setup.

Chapter 4

Branched Multi-Task Networks

In this chapter, we study branched multi-task networks as an alternative for existing encoder-focused models. Branched multi-task networks typically start with a number of shared layers, after which different tasks branch out into their own sequence of layers. Understandably, as the number of possible network configurations is combinatorially large, deciding what layers to share and where to branch out becomes cumbersome. Therefore, prior works have relied on ad hoc methods to determine the level of layer sharing, which is found suboptimal. We go beyond these limitations and propose an approach to automatically construct branched multi-task networks, by leveraging the employed tasks' affinities. Given a specific budget, i.e. number of learnable parameters, the proposed approach generates architectures, in which shallow layers are task-agnostic, whereas deeper ones gradually grow more task-specific. An extensive experimental analysis across numerous, diverse multi-tasking datasets shows that, for a given budget, our method consistently yields networks with the highest performance, while for a certain performance threshold it requires the least amount of learnable parameters.

The content of this chapter is based upon the following publication:

- *Vandenhende, S., Georgoulis, S., De Brabandere, B., & Van Gool, L. (2020). Branched Multi-Task Networks: Deciding What Layers To Share. The British Machine Vision Conference.*

4.1 Introduction

Consider the following observation: deep neural networks tend to learn hierarchical image representations [171]. The early layers tend to focus on more general low-level image features, such as edges, corners, etc., while the deeper layers tend to extract high-level information that is more task-specific. Motivated by this observation, branched MTL networks opt to learn a similar hierarchical encoding structure (e.g. [107, 52]). These ramified networks typically start with a number of shared layers, after which different (groups of) tasks branch out into their own sequence of layers. In doing so, the different branches gradually become more task-specific as we move to the deeper layers. This behaviour aligns well with the hierarchical representations learned by deep neural nets.

A significant challenge is to decide on the layers that need to be shared among tasks. In particular, the number of possible network configurations grows quickly with the number of tasks. As a result, a trial-and-error procedure to define the optimal architecture becomes unwieldy. Therefore, previous works opted for the simple strategy of sharing the initial layers in the network, after which all tasks branch out simultaneously. The point at which the branching occurs is usually determined ad hoc [73, 51, 139]. This situation hurts performance, as a suboptimal grouping of tasks can lead to the sharing of information between unrelated tasks, known as *negative transfer* [182].

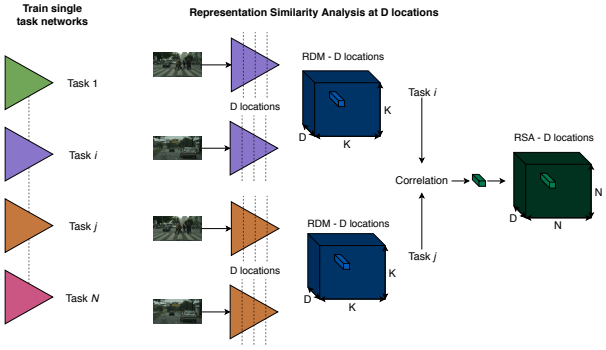
In this chapter, we go beyond the aforementioned limitations and propose a novel approach to decide on the degree of layer sharing between multiple visual recognition tasks in order to eliminate the need for manual exploration. To this end, we base the layer sharing on measurable levels of *task affinity* or *task relatedness*: two tasks are strongly related, if their single task models rely on a similar set of features. Zamir *et al.* [176] quantified this property by measuring the performance when solving a task using a variable sets of layers from a model pretrained on a different task. However, their approach is considerably expensive, as it scales quadratically with the number of tasks. Recently, [40] proposed a more efficient alternative that uses representation similarity analysis (RSA) to obtain a measure of task affinity, by computing correlations between models pretrained on different tasks. Given a dataset and a number of tasks, our approach uses RSA to assess the task affinity at arbitrary locations in a neural network. The task affinity scores are then used to construct a branched multi-task network in a fully automated manner. In particular, our task clustering algorithm groups similar tasks together in common branches, and separates dissimilar tasks by assigning them to different branches, thereby reducing the negative transfer between tasks. Additionally, our method allows to trade network complexity against task similarity. We provide extensive empirical evaluation of our method, showing its superiority in terms of multi-task performance versus computational resources.

4.2 Related Work

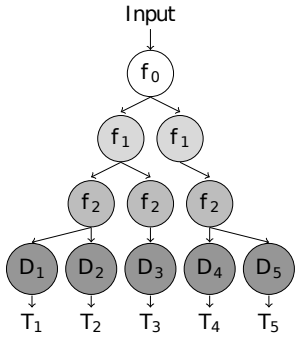
Fully-Adaptive Feature Sharing Our work bears some similarity to fully-adaptive feature sharing [107] (FAFS), which starts from a thin network where tasks initially share all layers, but the final one, and dynamically grows the model in a greedy layer-by-layer fashion. Task groupings, in this case, are decided on the probability of concurrently simple or difficult examples across tasks. Differently, (1) our method clusters tasks based on feature affinity scores, rather than example difficulty, which is shown to achieve better results for a variety of datasets; (2) the tree structure is determined offline using the precalculated affinities for the whole network, and not online in a greedy layer-by-layer fashion, which promotes task groupings that are optimal in a global, rather than local, sense.

Neural Architecture Search Neural architecture search (NAS) [43] aims to automate the construction of the network architecture. Different algorithms can be characterized based on their search space, search strategy or performance estimation strategy. Most existing works on NAS, however, are limited to task-specific models [187, 98, 125, 96, 131]. This is to be expected as when using NAS for MTL, layer sharing has to be jointly optimized with the layers types, their connectivity, etc., rendering the problem considerably expensive. To alleviate the heavy computation burden, a recent work [89] implemented an evolutionary architecture search for multi-task networks, while other researchers explored more viable alternatives, like routing [133], stochastic filter grouping [13], and feature partitioning [118]. In contrast to traditional NAS, the proposed methods do not build the architecture from scratch, but rather start from a predefined backbone network for which a layer sharing scheme is automatically determined.

Transfer Learning Transfer learning [121] uses the knowledge obtained when solving one task, and applies it to a different but related task. Our work is loosely related to transfer learning, as we use it to measure levels of task affinity. Zamir *et al.* [176] provided a taxonomy for task transfer learning to quantify such relationships. However, their approach scales unfavorably w.r.t. the number of tasks, and we opted for a more efficient alternative proposed by [40]. The latter uses RSA to obtain a measure of task affinity, by computing correlations between models pretrained on different tasks. In our method, we use the performance metric from their work to compare the usefulness of different feature sets for solving a particular task.



(a) Pipeline overview. (left) We train a single-task model for every task $t \in \mathcal{T}$. (middle) We use RSA to measure the task affinity at D predefined locations in the sharable encoder. In particular, we calculate the representation dissimilarity matrices (RDM) for the features at D locations using K images, which gives a $D \times K \times K$ tensor per task. (right) The affinity tensor $\mathbf{A}$ is found by calculating the correlation between the RDM matrices, which results in a three-dimensional tensor of size $D \times N \times N$, with N the number of tasks.



(b) Our pipeline's output is a branched multi-task network, similar to how NAS techniques output sample architectures. An example branched multi-task network is visualized here.

Figure 4.1: The proposed method: (a) calculate task affinities at various locations in the sharable encoder; (b) build a branched multi-task network based on the computed affinities.

4.3 Method

We aim to jointly solve N different visual recognition tasks $\mathcal{T} = \{t_1, \ldots, t_N\}$ given a computational budget $\mathcal{C}$, i.e. number of parameters or FLOPS. Consider a backbone architecture: an encoder, consisting of a sequence of shared layers or blocks f_l, followed by a decoder with a few task-specific layers. We assume an appropriate structure for layer sharing to take the shape of a tree. In particular, the first layers are shared by all tasks, while later layers gradually split off as they show more task-specific behavior. The proposed method aims to find an effective task grouping for the sharable layers f_l of the encoder, i.e. grouping related tasks together in the same branches of the tree. When two tasks are strongly related, we expect their single-task models to rely on a similar feature set [176]. Based on this viewpoint, the proposed method derives a task affinity score at various locations in the sharable encoder. The number of locations D can be freely determined as the number of candidate branching locations. As such, the resulting task affinity scores are used for the automated construction of a branched multi-task network that fits the computational budget $\mathcal{C}$. Figure 4.1 illustrates our pipeline, while Algorithm 1 summarizes the whole procedure.

4.3.1 Calculate Task Affinity Scores

As mentioned, we rely on RSA to measure task affinity scores. This technique has been widely adopted in the field of neuroscience to draw comparisons between behavioral models and brain activity (see [79]). Inspired by how [40] applied RSA to select tasks for transfer learning, we use the technique to assess the task affinity at predefined locations in the sharable encoder. Consequently, using the measured levels of task affinity, tasks are assigned in the same or different branches of a branched multi-task network, subject to the computational budget $\mathcal{C}$.

The procedure to calculate the task affinity scores is the following. As a first step, we train a single-task model for each task $t_i \in \mathcal{T}$. The single-task models use an identical encoder E - made of all sharable layers f_l - followed by a task-specific decoder D_{t_i}. The decoder contains only task-specific operations and is assumed to be significantly smaller in size compared to the encoder. As an example, consider jointly solving a classification and a dense prediction task. Some fully connected layers followed by a softmax operation are typically needed for the classification task, while an additional decoding step with some upscaling operations is required for the dense prediction task. Of course, the appropriate loss functions are applied in each case. Such operations are part of the task-specific decoder D_{t_i}. The different single-task networks are trained under the same conditions.

At the second step, we choose D locations in the sharable encoder E where we calculate a two-dimensional task affinity matrix of size $N \times N$. When concatenated, this results in a three-dimensional tensor $\mathbf{A}$ of size $D \times N \times N$ that holds the task affinities at the selected locations. To calculate these task affinities, we have to compare the representation dissimilarity matrices (RDM) of the single-task networks - trained in the previous step - at the specified D locations. To do this, a held-out subset of K images is required. The latter images serve to compare the dissimilarity of their feature representations in the single-task networks for every pair of images. Specifically, for every task t_i, we characterize these learned feature representations at the selected locations by filling a tensor of size $D \times K \times K$. This tensor contains the dissimilarity scores $1 - \rho$ between feature representations, with ρ the Pearson correlation coefficient. Specifically, $\mathbf{RDM}_{d,i,j}$ is found by calculating the dissimilarity score between the features at location d for image i and j. The 3-D tensors are linearized to 1-D tensors to calculate the pearson correlation coefficient.

For a specific location d in the network, the computed RDMs are symmetrical, with a diagonal of zeros. For every such location, we measure the similarity between the upper or lower triangular part of the RDMs belonging to the different single-task networks. We use the Spearman's correlation coefficient r_s to measure similarity. When repeated for every pair of tasks, at a specific location d, the result is a symmetrical matrix of size $N \times N$, with a diagonal of ones. Concatenating over the D locations in the sharable

Algorithm 1 Branched Multi-Task Networks - Task clustering

Input: Tasks $\mathcal{T}$, K images $\mathcal{I}$, a sharable encoder E with D locations where we can branch, a set of task specific decoders D_t and a computational budget $\mathcal{C}$.
for t in $\mathcal{T}$ **do**
    Train the encoder E and task-specific decoder D_t for task t.
    $RDM^t \leftarrow RDM(E, D, \mathcal{I})$ ▷ RDM for task t.
end for
$A_{d,i,j} \leftarrow r_s \left(\text{triu} \left(RDM^{t_i}_{d,:,:} \right), \text{triu} \left(RDM^{t_j}_{d,:,:} \right) \right)$ **for** t_i, t_j in $\mathcal{T}$ and d in locations
$D = 1 - A$ ▷ Task dissimilarity
Return: Task-grouping with minimal task dissimilarity that fits within $\mathcal{C}$

encoder, we end up with the desired task affinity tensor of size $D \times N \times N$. Note that, in contrast to prior work [107], the described method focuses on the features used to solve the single tasks, rather than the examples and how easy or hard they are across tasks, which is shown to result in better task groupings in Section 4.4. Furthermore, the computational overhead to determine the task affinity scores based on feature correlations is negligible. We conclude that the computational cost of the method boils down to pre-training N single task networks. A computational cost analysis can be found in the experiments section.

Other measures of task similarity [1, 31] probed the features from a network pre-trained on ImageNet. This avoids the need to pre-train a set of single-task networks first. However, in this case, the task dictionaries only consisted of various, related classification problems. Differently, we consider more diverse, and loosely related task (see Section 4.4). In our case, it is arguably more important to learn the task-specific information needed to solve a task. This motivates the use of pre-trained single-task networks.

4.3.2 Construct a Branched Multi-Task Network

Given a computational budget $\mathcal{C}$, we need to derive how the layers (or blocks) f_l in the sharable encoder E should be shared among the tasks in $\mathcal{T}$. Each layer $f_l \in E$ is represented as a node in the tree, i.e. the root node contains the first layer f_0, and nodes at depth l contain layer(s) f_l. The granularity of the layers f_l corresponds to the intervals at which we measure the task affinity in the sharable encoder, i.e. the D locations. When the encoder is split into b_l branches at depth l, this is equivalent to a node at depth l having b_l children. The leaves of the tree contain the task-specific decoders D_t. Figure 4.1b shows an example of such a tree using the aforementioned notation. Each node is responsible for solving a unique subset of tasks.

The branched multi-task network is built with the intention to separate dissimilar

tasks by assigning them to separate branches. To this end, we define the dissimilarity score between two tasks t_i and t_j at location d as $1 - \mathbf{A}_{d,i,j}$, with $\mathbf{A}$ the task affinity tensor[1]. The branched multi-task network is found by minimizing the sum of the task dissimilarity scores at every location in the sharable encoder. In contrast to prior work [107], the task affinity (and dissimilarity) scores are calculated a priori. This allows us to determine the task clustering offline. Since the number of tasks is finite, we can enumerate all possible trees that fall within the given computational budget $\mathcal{C}$. Finally, we select the tree that minimizes the task dissimilarity score. The task dissimilarity score of a tree is defined as $C_{cluster} = \sum_l C^l_{cluster}$, where $C^l_{cluster}$ is found by averaging the maximum distance between the dissimilarity scores of the elements in every cluster. The use of the maximum distance encourages the separation of dissimilar tasks. By taking into account the clustering cost at all depths, the procedure can find a task grouping that is considered optimal in a global sense. This is in contrast to the greedy approach in [107], which only minimizes the task dissimilarity locally, i.e. at isolated locations in the network.

4.4 Experiments

4.4.1 Cityscapes

Setup

We tackle the semantic segmentation, instance segmentation and monocular depth estimation tasks on Cityscapes. As in prior works [73, 139], we use a ResNet-50 encoder with dilated convolutions, followed by a Pyramid Spatial Pooling (PSP) [58] decoder. Every input image is rescaled to 512 x 256 pixels. We reuse the approach from [73] for the instance segmentation task, i.e. we consider the proxy task of regressing each pixel to the center of the instance it belongs to. The depth estimation is learned by placing an L1 regression loss on the disparity map.

Results

We measure the task affinity after every block (1 to 4) in the ResNet-50 model (see Figure 4.2). The task affinity decreases in the deeper layers, due to the features becoming more task-specific. We compare the performance of the task groupings generated by our method with those by other approaches.

[1] This is not to be confused with the dissimilarity score used to calculate the RDM elements $\mathbf{RDM}_{d,i,j}$.

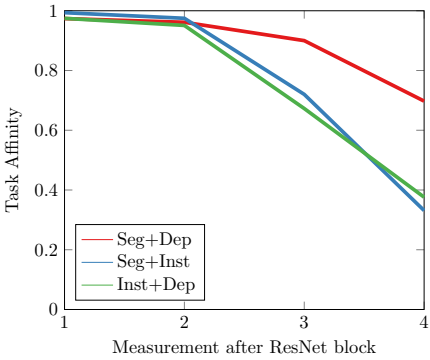


Figure 4.2: Task affinity scores measured after each residual block in a ResNet-50 model on Cityscapes.

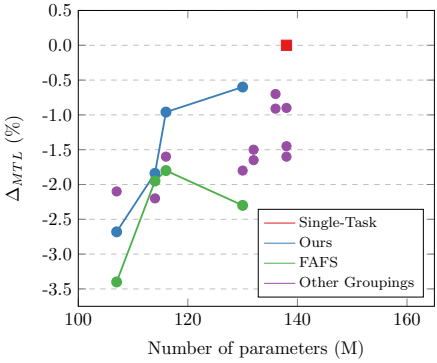


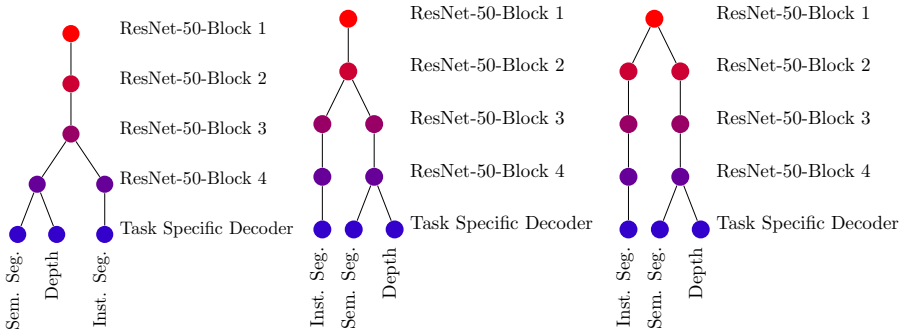
Figure 4.3: Number of parameters versus multi-task performance on Cityscapes for different task groupings. The 'Other Groupings' contain any remaining tree structures that can be found by randomly branching the model in the last three ResNet blocks.

We trained all possible task groupings that can be derived from branching the model in the last three ResNet blocks. Figure 4.3 visualizes performance versus number of parameters for the trained architectures. Depending on the available computational budget $\mathcal{C}$, our method generates a specific task grouping. We visualize these generated groupings as a blue path in Figure 4.3, when gradually increasing the computational budget $\mathcal{C}$. Similarly, we consider the task groupings when branching the model based on the task affinity measure proposed by FAFS [107] (green path). We find that, in comparison, the task groupings devised by our method achieve higher performance within a given computational budget $\mathcal{C}$. Furthermore, in the majority of cases, for a fixed budget $\mathcal{C}$ the proposed method is capable of selecting the best performing task grouping w.r.t. performance vs parameters metric (blue vs other).

We also compare our branched multi-task networks with cross-stitch networks [114], NDDR-CNNs [47] and MTAN [101] in Table 4.1. While cross-stitch nets and NDDR-CNNs give higher multi-task performance, attributed to their computationally expensive soft parameter sharing setting, our branched networks can strike a better trade-off between the performance and number of parameters. In particular, we can effectively sample architectures which lie between the extremes of a baseline multi-task model and a cross-stitch or NDDR-CNN architecture. Finally, our models provide a more computationally efficient alternative to the MTAN model, which reports similar performance while using more parameters. We visualize our task groupings from

Table 4.1: Results on the Cityscapes validation set.

Method	**Sem. Seg.** (IoU) ↑	**Inst. Seg.** (px) ↓	**Disparity** (px) ↓	**# Params** (M) ↓	Δ_{MTL} (%)↑
Single task	65.2	11.7	2.57	138	+0.00
MTL baseline	61.5	11.8	2.66	92	-3.33
MTAN	62.8	11.8	2.66	113	-2.53
Cross-stitch	65.1	11.6	2.55	140	+0.42
NDDR-CNN	65.6	11.6	2.54	190	+0.89
Ours - 1	62.1	11.7	2.66	107	-2.68
Ours - 2	62.7	11.7	2.62	114	-1.84
Ours - 3	64.1	11.6	2.62	116	-0.96



(a) Ours - 1 (b) Ours - 2 (c) Ours - 3

Figure 4.4: Task groupings generated by our method on the Cityscapes dataset.

Table 4.1 in Figure 4.4.

4.4.2 Taskonomy

Setup

We tackle the scene categorization (C), semantic segmentation (S), edge detection (E), monocular depth estimation (D) and keypoint detection (K) tasks. All input images were rescaled to 256 x 256 pixels. We use a ResNet-50 encoder and replace the last stride 2 convolution by a stride 1 convolution. A 15-layer fully-convolutional decoder is used for the pixel-to-pixel prediction tasks. The decoder is composed of five convolutional layers followed by alternating convolutional and transposed convolutional layers. We

Table 4.2: Results on the tiny Taskonomy test set. The results for edge (E) and keypoints (K) detection were multiplied by a factor of 100 for better readability. The FAFS models refer to generating the task groupings with the task affinity technique proposed by [107].

Method	**Depth** (L1) ↓	**Sem. Seg** (IoU) ↑	**Categ.** (Top-5) ↑	**Edge** (L1) ↓	**Keyp.** (L1) ↓	**# Params** (M) ↓	$\mathbf{\Delta_{MTL}}$ (%) ↑
Single task	0.60	43.5	66.0	0.99	0.23	224	+ 0.00
MTL baseline	0.75	47.8	56.0	1.37	0.34	130	- 22.50
MTAN	0.71	43.8	59.6	1.86	0.40	158	-37.36
Cross-stitch	0.61	44.0	58.2	1.35	0.50	224	- 32.29
NDDR-CNN	0.66	45.9	64.5	1.05	0.45	258	- 21.02
FAFS - 1	0.74	46.1	62.7	1.30	0.39	174	- 24.5
FAFS - 2	0.80	39.9	62.4	1.68	0.52	188	- 48.32
FAFS - 3	0.74	46.1	64.9	1.05	0.27	196	- 8.48
Ours - 1	0.76	47.6	63.3	1.12	0.29	174	- 11.88
Ours - 2	0.74	48.0	63.6	0.96	0.35	188	- 12.66
Ours - 3	0.74	47.9	64.5	0.94	0.26	196	- 4.93

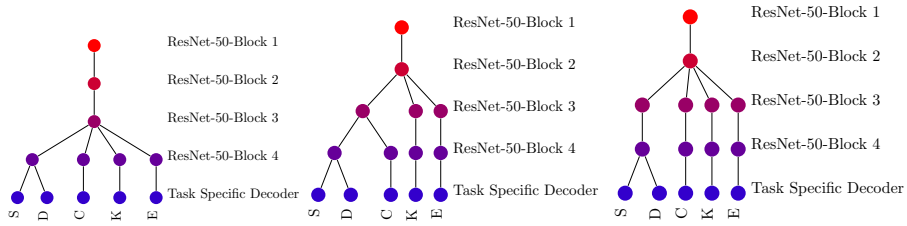
use ReLU as non-linearity. Batch normalization is included in every layer except for the output layer. We use Kaiming He's initialization for both encoder and decoder. We use an L1 loss for the depth (D), edge detection (E) and keypoint detection (K) tasks. The scene categorization task is learned with a KL-divergence loss. We report performance on the scene categorization task by measuring the overlap in top-5 classes between the predictions and ground truth.

The multi-task models were optimized with task weights $w_s = 1, w_d = 1, w_k = 10, w_e = 10$ and $w_c = 1$. The ground-truth heatmaps were linearly rescaled to lie between 0 and 1. During training we normalize the depth map by the standard deviation.

Results

The task affinity is again measured after every ResNet block. Since the number of tasks increased to five, it is very expensive to train all task groupings exhaustively, as done above. Instead, we limit ourselves to three architectures that are generated when gradually increasing the parameter budget. As before, we compare our task groupings against the method from [107]. The numerical results can be found in Table 4.2. The task groupings themselves are shown in Figure 4.5.

The effect of the employed task grouping technique can be seen from comparing the performance of our models against the corresponding FAFS models, generated



(a) Ours - 1 (b) Ours - 2 (c) Ours - 3

Figure 4.5: Task groupings generated by our method on the Taskonomy dataset.

by [107]. The latter are consistently outperformed by our models. Compared to the results on Cityscapes (Figure 4.3), we find that the multi-task performance is much more susceptible to the employed task groupings, possibly due to negative transfer. Furthermore, we observe that none of the soft parameter sharing models can handle the larger, more diverse task dictionary: the performance decreases when using these models, while the number of parameters increases. This is in contrast to our branched multi-task networks, which seem to handle the diverse set of tasks rather positively. As opposed to [176], but in accordance with [110], we show that it is possible to solve many heterogeneous tasks simultaneously when the negative transfer is limited, by separating dissimilar tasks from each other in our case. In fact, our approach is the first to show such consistent performance across different multi-tasking scenarios and datasets. Existing approaches seem to be tailored for particular cases, e.g. few/correlated tasks, synthetic-like data, binary classification only tasks, etc., whereas we show stable performance across the board of different experimental setups.

4.4.3 CelebA

Setup

The CelebA dataset [102] contains over 200k images of celebrities, labeled with 40 facial attribute categories. We treat the prediction of each facial attribute as a single binary classification task, as in [107, 139, 66]. To ensure a fair comparison: we reuse the thin-ω model from [107] in our experiments; the parameter budget $\mathcal{C}$ is set for the model to have the same amount of parameters as prior work. The CNN architecture is based on the VGG-16 model [141]. The number of convolutional features is set to the minimum between ω and the width of the corresponding layer in the VGG-16 model. The fully connected layers contain $2 \cdot \omega$ features. The loss function is a sigmoid cross-entropy loss with uniform weighing scheme.

Table 4.3: Results on the CelebA test set. The **Ours-32**, *Ours-64* architectures are found by optimizing the task clustering for the parameter budget that is used in the **FAFS**, *GNAS* model respectively.

Method	**Acc.** (%)	**#Params** (M) ↓
LNet+ANet [156]	87	-
Walk and Learn [156]	88	-
MOON [135]	90.94	119.73
Independent Group [55]	91.06	-
MCNN [55]	91.26	-
MCNN-AUX [55]	91.29	-
VGG-16 [107]	91.44	134.41
FAFS [107]	**90.79**	**2.09**
GNAS [55]	*91.63*	*7.73*
Res-18 (Uniform) [139]	90.38	11.2
Res-18 (MGDA) [139]	91.75	11.2
Ours-32	**91.46**	**2.20**
Ours-64	*91.73*	*7.73*

Results

Table 4.3 shows the results on the CelebA test set. Our branched multi-task networks outperform earlier works [107, 66] when using a similar amount of parameters. Since the Ours-32 model (i.e. ω is 32) only differs from the FAFS model on the employed task grouping technique, we can conclude that the proposed method devises more effective task groupings for the attribute classification tasks on CelebA. Furthermore, the Ours-32 model performs on par with the VGG-16 model, while using 64 times less parameters. Finally, we show a visualization of the tree structure employed by the Ours-32 model in Figure 4.6.

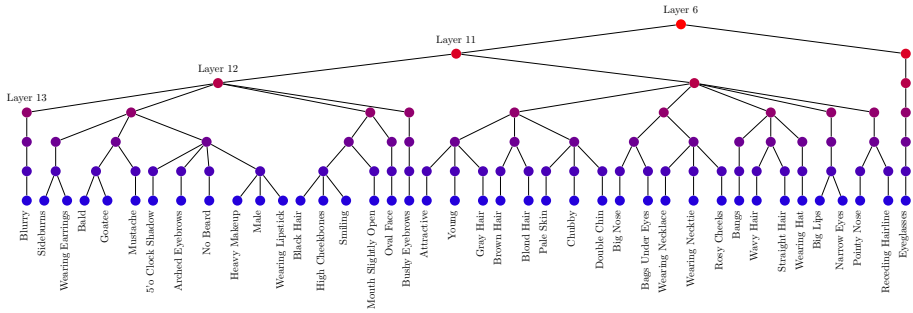


Figure 4.6: Grouping of 40 person attribute classification tasks on CelebA in a thin VGG-16 architecture (Ours-32).

4.4.4 Computational Analysis

We provide an analysis to identify the computational costs related to the different steps when calculating the task affinity scores. The following three steps can be identified:

- Train N single task networks. It is possible to use a subset of the available training data to reduce the training time. We verified that using a random subset of 500 training images (1/5) on Cityscapes results in the same task groupings.
- Compute the RDM matrix for all N networks at D pre-determined layers. This requires to compute the features for a set of K images at the D pre-determined layers in all N networks. The K images are usually taken as held-out images from the train set. We used $K = 500$ in our experiments. In practice this means that computing the image features comes down to evaluating every model on K images. The computed features are stored on disk afterwards. The RDM matrices are calculated from the stored features. This requires to calculate $N \times D \times K \times K$ correlations between two feature vectors (can be performed in parallel). In practice, the computation time is negligible compared to training the single-task networks.
- Compute the RSA matrix at D locations for all N tasks. This requires to calculate $D \times N \times N$ correlations between the lower triangle part of the $K \times K$ RDM matrices. Again, the computation time can be neglected.

In conclusion, the computational cost of our method boils down to training N single task networks plus some small overhead. Notice that cross-stitch networks [114] and NDDR-CNNs [47] also pre-train a set of single-task networks first, before combining them together using a soft parameter sharing mechanism. Differently, our method only

requires to use a subset of the training images. We conclude that our method does not suffer from overhead compared to these methods.

4.5 Conclusion

We introduced a principled approach to automatically construct branched multi-task networks for a given computational budget. To this end, we leverage the employed tasks' affinities as a quantifiable measure for layer sharing. The proposed approach can be seen as an abstraction of NAS for MTL, where only layer sharing is optimized, without having to jointly optimize the layers types, their connectivity, etc., as done in traditional NAS, which would render the problem considerably expensive. Extensive experimental analysis shows that our method outperforms existing ones w.r.t. the important metric of multi-tasking performance vs number of parameters, while at the same time showing consistent results across a diverse set of multi-tasking scenarios and datasets.

Limitations Branched Multi-Task Networks allow to strike a better trade-off between the performance and the used amount of computational resources. This allows to significantly reduce the size of the model w.r.t. single-task learning. Unfortunately, we do not observe similar gains for the performance. Therefore, in the next chapter, we introduce a novel architecture specifically designed for tackling multiple dense prediction tasks. The resulting model does deliver on the full promise of multi-task learning, i.e. improved performance, smaller memory footprint and reduced number of FLOPS.

Extensions/Follow-Up Work Different from our work, **Branched Multi-Task Architecture Search** [14] and **Learning To Branch** [52] have directly optimized the network topology without relying on pre-computed task relatedness scores. More specifically, they rely on a tree-structured network design space where the branching points are casted as a Gumbel softmax operation. This strategy has the advantage over our work that the task groupings can be directly optimized end-to-end for the tasks under consideration. Moreover, both methods can easily be applied to any set of tasks, including both image classification and per-pixel prediction tasks. Similar to [107, 152], a compact network topology can be obtained by including a resource-aware loss term. In this case, the computational budget is jointly optimized with the multi-task learning objective in an end-to-end fashion.

Chapter 5

Multi-Scale Multi-Task Interaction Networks

In this chapter, we revisit the decoder-focused approaches. First, we argue about the importance of considering task interactions at multiple scales when distilling task information in a multi-task learning setup. In contrast to common belief, our experimental analysis shows that tasks with high affinity at a certain scale are not guaranteed to retain this behavior at other scales, and vice versa.

Next, we propose a novel architecture, namely MTI-Net, that builds upon our findings in three ways. First, MTI-Net explicitly models task interactions at every scale via a multi-scale multi-modal distillation unit. Second, MTI-Net propagates distilled task information from lower to higher scales via a feature propagation module. Third, MTI-Net aggregates the refined task features from all scales via a feature aggregation unit to produce the final per-task predictions.

Finally, extensive experiments on two multi-task dense labeling datasets show that, unlike prior work, our multi-task model delivers on the full potential of multi-task learning, that is, smaller memory footprint, reduced number of calculations, and better performance w.r.t. single-task learning.

The content of this chapter is based upon the following publication:

- *Vandenhende, S., Georgoulis, S., & Van Gool, L. (2020). MTI-Net: Multi-scale task interaction networks for multi-task learning. European Conference on Computer Vision.*

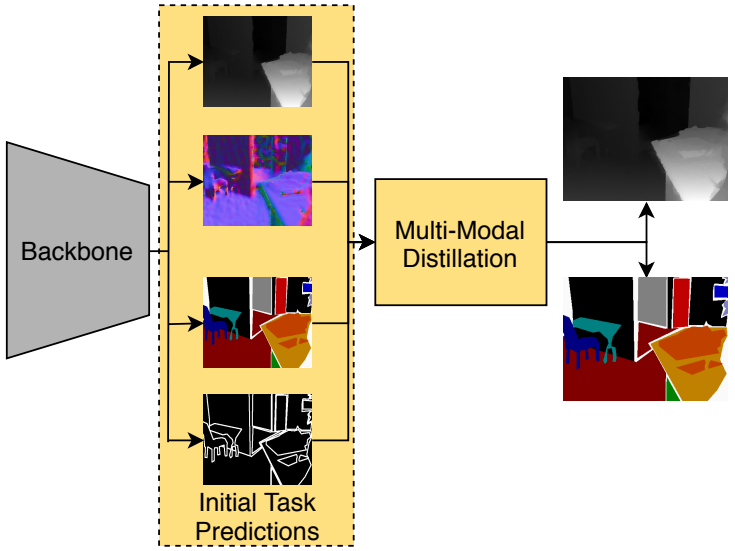


Figure 5.1: An overview of the architecture used in PAD-Net [165] and PAP-Net [179]. Features extracted from a backbone network are used to make initial task predictions. The task features are combined through a distillation unit before making the final task predictions.

5.1 Motivation

First, we introduce the necessary background materials. Next, we motivate the use of a decoder-focused model that distills task information at multiple scales.

5.1.1 Decoder-Focused Approaches

Decoder-focused approaches [165, 178, 179, 186] try to explicitly distill information from other tasks, as a complementary signal to improve task performance. Typically, this is achieved by combining an existing backbone network with a multi-step decoding process.

In more detail, the shared features of the backbone network are processed by a set of task-specific heads, that produce an initial prediction for every task. We further refer to the backbone and the task-specific heads as the *front-end* of the network. The task-specific heads produce a per-task feature representation of the scene that is more task-aware than the shared features of the backbone network. The information from these task-specific feature representations is then combined via a multi-modal

distillation unit, before making the final task predictions. As shown in Figure 5.1, it is possible that some tasks are only predicted in the front-end of the network. The latter are known as auxiliary tasks, since they serve as proxies in order to improve the performance on the final tasks. We refer the interested reader to Section 3.1.3 for an in-depth discussion on existing decoder-focused approaches.

For brevity, we adopt the following notations. *Backbone features*: the shared features (at the last layer) of the backbone network; *Task features*: the task-specific feature representations (at the last layer) of each task-specific head; *Distilled task features*: the task features after multi-modal distillation; *Initial task predictions*: the per-task predictions at the front-end of the network; *Final task predictions*: the network outputs. Note that, backbone features, task features and distilled task features can be defined at a single scale or multiple scales.

5.1.2 Task interactions at different scales

Existing decoder-focused approaches follow a common pattern: they perform multi-modal distillation at a fixed scale, i.e. the backbone features. This rests on the assumption that all relevant task interactions can solely be modeled through a single filter operation with specific receptive field. For all we know, this is not always the case. In fact, tasks can influence each other differently for different receptive field sizes. Consider, for example, Figure 5.2. The local patches in the depth map provide little information about the semantics of the scene. However, when we enlarge the receptive field, the depth map reveals a person's shape, hinting at the scene's semantics. Note that the local patches can still provide valuable information, e.g. to improve the local alignment of edges between tasks.

To quantify the degree to which tasks share common local structures w.r.t. the size of the receptive field, we conduct the following experiment. We measure the pixel affinity in local patches on the label space of each task, using kernels of fixed size. The size of the receptive field can be selected by choosing the dilation for the kernel. We consider the tasks of semantic segmentation, depth estimation and edge detection on the NYUD dataset. A pair of semantic pixels is considered similar when both pixels belong to the same category. For the depth estimation task, we threshold the relative difference between pairs of pixels; pixels below the threshold are similar. Once the pixel affinities are calculated for every task, we measure how well similar and dissimilar pairs are matched across tasks. We repeat this experiment using different dilations for the kernel, effectively changing the receptive field. Figure 5.3 illustrates the result.

A first observation is that affinity patterns are matched well across tasks, with up to 65% of pair correspondence in some cases. This indicates that different tasks can share common structures in parts of the image. A second observation is that the degree to

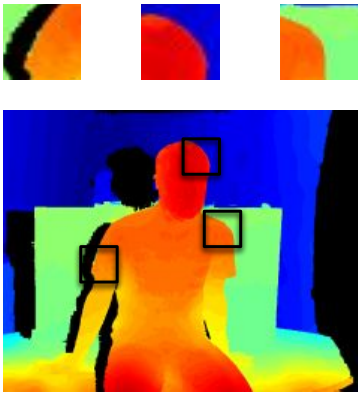


Figure 5.2: Three local patches from a depth map. Depending on the patch size, i.e. receptive field, depth information can be utilized differently by other tasks, e.g. semantic segmentation and edges.

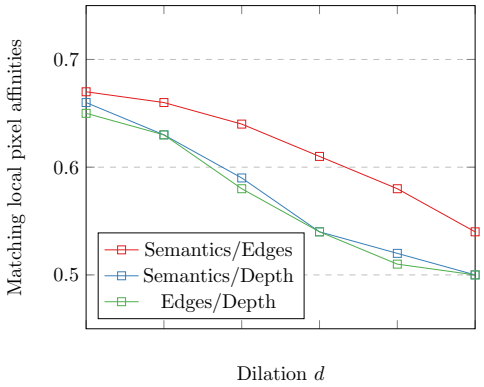


Figure 5.3: To quantify task interactions w.r.t. scale, pixel affinities on the label space of each task, as defined in [179], are calculated in local patches. The correspondences in the affinity patterns between tasks are plotted as a function of the patch size, i.e. kernel dilation.

which the affinity patterns are matched across tasks is dependent on the receptive field, which in turn, corresponds to the used dilation. This validates our initial assumption that the statistics of task interactions do not always remain constant, but rather depend on the scale, i.e. receptive field. Based on these findings, in the next section we introduce a model that distills information from different tasks at multiple scales. By doing so, we are able to capture the unique task interactions at each individual scale, overcoming the limitations of existing decoder-focused models.

5.2 Method

5.2.1 Multi-scale multi-modal distillation

We propose a multi-task architecture that explicitly takes into account task interactions at multiple scales. Figure 5.4 shows a visualization of our model. First, an off-the-shelf backbone network extracts a multi-scale feature representation from the input image. Such multi-scale feature extractors have been used in semantic segmentation [132, 157, 75], object detection [92, 157], pose estimation [119, 143], etc. In Section 5.3 we verify our approach using two such backbones, i.e. HRNet [157] and FPN [92], but any multi-scale feature extractor can be used instead.

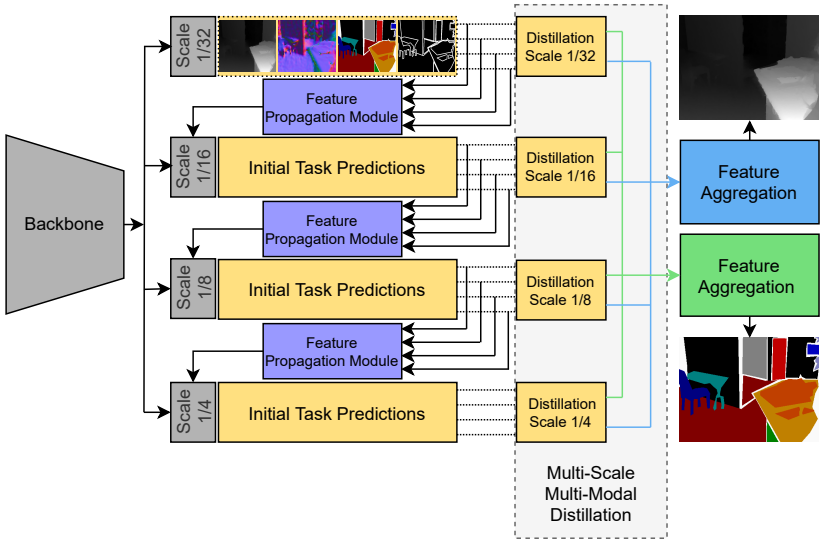


Figure 5.4: **Overview of the proposed MTI-Net**. Starting from a backbone that extracts multi-scale features, initial task predictions are made at each scale. The task features are distilled separately at every scale, allowing our model to capture task interactions at multiple scales, i.e. receptive fields. After distillation, the distilled task features from all scales are aggregated to make the final task predictions. To boost performance, we extend our model with a feature propagation mechanism that passes distilled information from lower resolution task features to higher ones.

From the multi-scale feature representation (i.e. backbone features) we make initial task predictions at each scale. These initial task predictions at a particular scale are found by applying a set of task-specific heads to the backbone features extracted at that scale. The result is a per-task representation of the scene (i.e. task features) at a multitude of scales. Not only does this add deep supervision to our network, but the task features can now be distilled at each scale separately. This allows us to have multiple task interactions, each modeled for a specific receptive field size, as proposed in Section 5.1.2.

Next, we refine the task features by distilling information from the other tasks using a spatial attention mechanism [165]. Yet, our multi-modal distillation process is repeated at each scale, i.e. we apply multi-scale, multi-modal distillation. The distilled task features $F^o_{k,s}$ for task k at scale s are found as:

$$F^o_{k,s} = F^i_{k,s} + \sum_{l \neq k} \sigma\left(W_{k,l,s} F^i_{l,s}\right) \odot \left(W'_{k,l,s} F^i_{l,s}\right), \tag{5.1}$$

where $\sigma\left(W_{k,l,s}F^i_{l,s}\right)$ returns a per-scale spatial attention mask, that is applied to the task features $F^i_{l,s}$ from task l at scale s. Note that our approach is not necessarily limited to the use of spatial attention, but any type of feature distillation (e.g. squeeze-and-excitation [63]) can easily be plugged in. Through repetition, we calculate the distilled task features at every scale. As the bulk of filter operations is performed on low resolution feature maps, the computational overhead of our model is limited. We make a detailed resource analysis in Section 5.3.

5.2.2 Feature propagation across scales

In Section 5.2.1 actions at each scale were performed in isolation. To sum up, we made initial task predictions at each scale, from which we refined the task features through multi-modal distillation at each individual scale separately. However, as the higher resolution scales have a limited receptive field, the front-end of the network could have a hard time to make good initial task predictions at these scales, which in turn, would lead to low quality task features there. To remedy this situation we introduce a feature propagation mechanism, where the backbone features of a higher resolution scale are concatenated with the task features from the preceding lower resolution scale, before feeding them to the task-specific heads of the higher resolution scale to get the task features there.

A trivial implementation for our Feature Propagation Module (FPM) would be to just upsample the task features from the previous scale and pass them to the next scale. We opt for a different approach however, and design the FPM to behave similarly to the multi-modal distillation unit of Section 5.2.1, in order to model task interactions at this stage too. Figure 5.5 gives an overview of our FPM. We first use a *feature harmonization* block to combine the task features from the previous scale to a shared representation. We then use this shared representation to refine the task features from the previous scale, before passing them to the next scale. The refinement happens by selecting relevant information from the shared representation through a *squeeze-and-excitation* block [63]. Note that, since we refine the features from a single shared representation, instead of processing each task independently as done in the multi-modal distillation unit of Section 5.2.1, the computational cost is significantly smaller.

Feature harmonization. The FPM receives as input the task features from N tasks of shape $C \times H \times W$. Our feature harmonization module combines the received task features into a shared representation. In particular, the set of N task features is first concatenated and processed by a learnable non-linear function f. The output is split into N chunks along the channel dimension, that match the original number of channels C. We then apply a softmax function along the task dimension to generate a task attention mask. The attended features are concatenated and further processed to

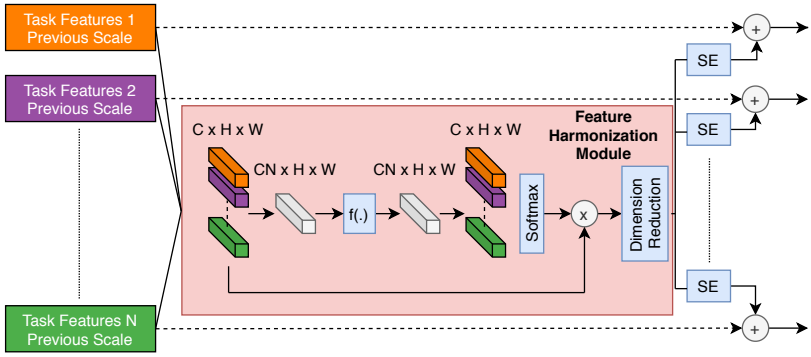


Figure 5.5: **Overview of the proposed Feature Propagation Module (FPM).** First, task features from a lower scale are concatenated and mapped to a shared representation by the feature harmonization module. The task features are then refined by extracting information from the shared representation through squeeze-and-excitation (SE) [63], and are added as a residual to the original ones. Finally, these refined task features will be concatenated with the backbone features of the preceding higher scale.

reduce the number of channels from $N \cdot C$ to C. The output is a shared representation based on information from all tasks.

Refinement through Squeeze-And-Excitation. The use of a shared representation can degrade performance when tasks are unrelated. We resolve this situation by applying a per-task channel gating function to the shared representation. This effectively allows each task to select the relevant features from the shared representation. The channel gating mechanism is implemented here as a squeeze-and-excitation (SE) block [63]. This is due to the fact that SE has shown great potential in MTL (e.g. [110]), yet other gating mechanisms could be used instead. After applying the SE module, the refined task features are added as a residual to the original task features.

5.2.3 Feature aggregation

The multi-scale, multi-modal distillation described in Section 5.2.1 results in distilled task features at every scale. The latter are upsampled to the highest scale and concatenated, resulting in a final feature representation for every task. The final task predictions are found by decoding these final feature representations by task-specific heads again. All implementation details are discussed in Section 5.3. It is worth mentioning that our model has the possibility to add auxiliary tasks in the front-end of the network, similar to PAD-Net [165]. In our case however, the auxiliary tasks are predicted at multiple scales.

5.2.4 Optimization

The model is trained in an end-to-end fashion. Assume we make initial task predictions at scales $\mathcal{S}$. Further, let $\mathcal{T}$ denote the set of main tasks and $\mathcal{A}$ denote the set of auxiliary tasks. We use the notation $\mathcal{L}_t^s$ for the loss of the initial predictions of task t at scale s, and $\mathcal{L}_t^o$ for the loss of the final predictions of task t. Further, we define the task-specific weight for task t as w_t. Note, we adopt the same task-specific weights for the initial and final task predictions. The optimization objective can be written as

$$\mathcal{L}_{total} = \sum_{t \in \mathcal{T}} \sum_{s \in \mathcal{S}} w_t \cdot \mathcal{L}_t^s + \sum_{a \in \mathcal{A}} \sum_{s \in \mathcal{S}} w_a \cdot \mathcal{L}_a^s + \sum_{t \in \mathcal{T}} w_t \cdot \mathcal{L}_t^o. \tag{5.2}$$

5.3 Experiments

5.3.1 Setup

Datasets & Implementation Details

We perform our experimental evaluation on the PASCAL [45] and NYUD [140] datasets (see Section 2.2). We build our approach on top of two different backbone networks, i.e. FPN [92] and HRNet [143]. We use the different output scales of the selected backbone networks to perform multi-scale operations. This translates to four scales (1/4, 1/8, 1/16, 1/32). The task-specific heads are implemented as two basic residual blocks [59]. All our experiments are conducted with pre-trained ImageNet weights.

We use the L1 loss for depth estimation and the cross-entropy loss for semantic segmentation on NYUD. As in prior work [76, 109, 110], the edge detection task is trained with a positively weighted $w_{pos} = 0.95$ binary cross-entropy loss. We do not adopt a particular loss weighing strategy on NYUD, but simply sum the losses together. On PASCAL, we reuse the training setup from [110] to facilitate a fair comparison. We reuse the loss weights from there. The initial task predictions in the front-end of the network use the same loss weighing as the final task predictions. In contrast to [165, 179, 178], we do not use a two-step training procedure where the front-end is pre-trained separately. Instead, we simply train the complete architecture end-to-end.

Evaluation Metrics

We evaluate the performance of the backbone networks on the single tasks first. The optimal dataset F-measure (*odsF*) [112] is used to evaluate the edge detection task. The semantic segmentation, saliency estimation and human part segmentation tasks are

evaluated using mean intersection over union (*mIoU*). We use the mean error (*mErr*) in the predicted angles to evaluate the surface normals. The depth estimation task is evaluated using the root mean square error (*rmse*). The multi-task learning performance Δ_{MTL} is defined in Equation 2.2.

Baselines

On NYUD, we compare MTI-Net against the state-of-the-art PAD-Net [165]. PAD-Net was originally designed for a single scale, but it is easy to plug-in a multi-scale backbone network and directly compare the two approaches. In contrast, a comparison with [178] is not possible, as this work was strictly designed for a pair of tasks, without any straightforward extension to the MTL setting. Finally, PAP-Net [179] adopts an architecture that is similar to PAD-Net, but the multi-modal distillation is performed recursively on the feature affinities. We chose to draw the comparison with the more generic PAD-Net, since it performs on par with PAP-Net (see Section 5.3.3).

On PASCAL, we compare our method against the state-of-the-art ASTMT [110]. Note that a direct comparison with ASTMT is also not straightforward, as this model is by design single-scale and heavily based on a DeepLab-v3+ (DLv3+) backbone network that contains dilated convolutions. Due to the latter, simply plugging the same DLv3+ backbone into MTI-Net would break the multi-scale features required to uniquely model the task interactions at a multitude of scales. Yet, we provide a fair comparison with ASTMT by combining it with a ResNet-50 FPN backbone, to show that it is not just using a multi-scale backbone that leads to improved results.

5.3.2 Ablation Studies

Network components

In Table 5.1 we visualize the results of our ablation studies on NYUD and PASCAL with an HRNet18 backbone to verify how different components of our model contribute to the multi-task improvements.

We focus on the smaller NYUD dataset first (see Table 5.1a), that contains arguably related tasks. These are semantic segmentation (Seg) and depth prediction (Depth) as main tasks, edge detection (E) and surface normals (N) as auxiliary tasks. The MTL baseline (i.e. a shared encoder with task-specific heads) has lower performance (-1.71%) than the single-task models. This is inline with prior work [152, 110]. PAD-Net retains performance over the set of single-task models (-0.02%), and improves when adding the auxiliary tasks ($+0.52\%$). Using our model without the FPM between scales further improves the results (w/o auxiliary tasks: $+3.85\%$, w/ auxiliary tasks:

Table 5.1: Ablation studies on (a) NYUD and (b) PASCAL using an HRNet-18 backbone network. Auxiliary tasks are indicated between brackets.

(a) Results on NYUD.

Method	**Seg** ↑	**Depth** ↓	$\mathbf{\Delta_m}\%$ ↑
Single task	33.18	0.667	+ 0.00
MTL	32.09	0.668	- 1.71
PAD-Net	32.80	0.660	- 0.02
PAD-Net (N)	33.85	0.658	+ 1.65
PAD-Net (N+E)	32.92	0.655	+ 0.52
Ours (w/o FPM)	34.38	0.640	+ 3.85
Ours (w/o FPM) (N)	34.49	0.642	+ 3.84
Ours (w/o FPM) (N+E)	34.68	0.637	+ 4.48
Ours (w/ FPM)	35.12	0.620	+ 6.40
Ours (w/ FPM) (N)	36.22	**0.600**	+ 9.57
Ours (w/ FPM) (N+E)	**37.49**	0.607	**+ 10.91**

(b) Results on PASCAL.

Method	**Seg** ↑	**Parts** ↑	**Sal** ↑	**Edge** ↑	**Norm** ↓	$\mathbf{\Delta_m}\%$ ↑
Single task	60.07	60.74	67.18	69.70	14.59	+ 0.00
MTL (s)	54.53	59.54	65.60	-	-	- 4.26
MTL (a)	53.60	58.45	65.13	70.60	15.08	- 3.70
Ours (s)	64.06	62.39	68.09	-	-	+ 3.35
Ours (s)(E)	64.98	62.90	67.84	-	-	+ 3.98
Ours (s)(N)	63.74	61.75	67.90	-	-	+ 2.69
Ours (s)(E+N)	64.33	62.33	68.00	-	-	+ 3.36
Ours (a)	64.27	62.06	68.00	73.40	14.75	+ 2.74

$+4.48\%$). When including the FPM another significant boost in performance is achieved ($+6.40\%$). Further adding the auxiliary tasks can help to improve the quality of our predictions ($+10.91\%$).

Table 5.1b shows the ablation on PASCAL. We discriminate between a *small set (s)* and a *complete set (a)* of tasks. The small set contains the high-level (semantic and human parts segmentation) and mid-level (saliency) vision tasks. The complete set also adds the low-level (edges and normals) vision tasks. The MTL baseline leads to decreased performance, -4.26% and -3.70% on the small and complete set respectively. Instead, our model improves over the set of single-task models ($+3.35\%$) on the small task set (s), where we obtain solid improvements on all tasks. We also report the influence of adding additional auxiliary tasks to the front-end of the network. Adding edges

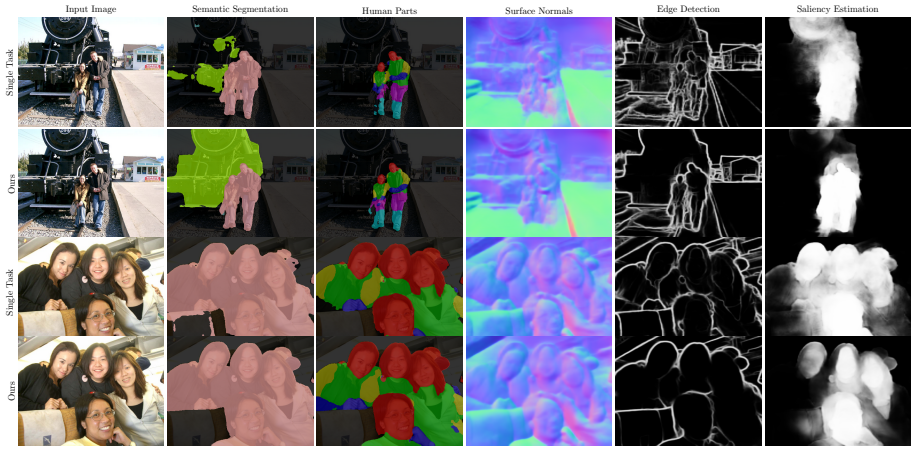


Figure 5.6: **Qualitative results on PASCAL.** We compare the predictions made by a set of single-task models (first row for every image) against the predictions made by our MTI-Net (second row for every image). Differences can be seen for semantic segmentation, edge detection and saliency estimation.

improves the multi-task performance to 3.98%, adding normals slightly decreases it to $+2.69\%$, while adding both keeps it stable $+3.36\%$. Finally, when learning all five tasks together, our model outperforms ($+2.74\%$) the set of single-task models. In general, all tasks gain significantly, except for normals, where we observe a small decrease in performance. We argue that this is due to the inevitable negative transfer that characterizes all models with shared operations (also [165, 110, 179]). Yet, to the best of our knowledge, this is the first work to not only report overall improved multi-task performance, but also to maximize the gains over the single-task models, when jointly predicting an increasing and diverse set of tasks. We refer to Figure 5.6 for qualitative results obtained with an HRNet-18 backbone.

Influence of scales

So far, our experiments included all four scales of the backbone network (1/4, 1/8, 1/16, 1/32). Here, we study the influence of using a different number of scales for the backbone. Table 5.2 summarizes this ablation on NYUD. Note that the use of a single scale (1/4) reduces our model to a PAD-Net like architecture. Using an increasing number of scales (1/4 vs + 1/8 vs + 1/16, ...) gradually improves performance. The results confirm our hypothesis from Section 5.1.2, i.e. task interactions should be modeled at multiple scales.

Table 5.2: Influence of using a different number of scales for the backbone network on NYUD.

Method	**Seg** ↑	**Depth** ↓	$\mathbf{\Delta_m}\%$ ↑
ST	33.18	0.667	+ 0.00
1/4 (Pad-Net)	32.80	0.660	- 0.02
1/4, 1/8	34.88	0.650	+ 3.80
1/4, 1/8, 1/16	35.01	0.630	+ 5.53
1/4, 1/8, 1/16, 1/32 (Ours)	35.12	0.620	+ 6.40

Table 5.3: Ablating the information flow within the proposed MTI-Net model on NYUD.

Method	**Seg** ↑	**Depth** ↓	$\mathbf{\Delta_m}\%$ ↑
ST	33.18	0.667	+ 0.00
Front-end @ 1/32 scale	32.02	0.670	- 1.87
Front-end @ 1/16 scale	33.02	0.660	+ 0.02
Front-end @ 1/8 scale	33.67	0.640	+ 2.72
Front-end @ 1/4 scale	34.05	0.633	+ 3.78
Final output	35.12	0.620	+ 6.40

Information flow

To quantify the flow of information, we measure the performance of the initial task predictions at different locations in the front-end of the network. Table 5.3 illustrates the results on NYUD. We observe that the performance gradually increases at the higher scales, due to the information that is being propagated from the lower scales via the FPM. The final prediction after aggregating the information from all scales is further improved substantially.

5.3.3 State-Of-The-Art Comparison

Comparison on PASCAL

Table 5.4 visualizes the comparison of our model against ASTMT and PAD-Net on PASCAL. We report the multi-tasking performance both w.r.t. the single-task models using the same backbone (ST) and the single-task models based on the R50-FPN backbone. As explained, in the only possible fair comparison, i.e. when using the same R50-FPN backbone, our model achieves higher multi-tasking performance compared to ASTMT ($+1.32\%$ vs -0.74%). Yet, as ASTMT is by design single-scale and

Table 5.4: Comparison with the state-of-the-art on PASCAL. Δ_{MTL} is reported for a set of single-tasking models using the same backbone.

Method	**Backbone**	**Seg** ↑	**Parts** ↑	**Sal** ↑	**Edge** ↑	**Norm** ↓	$\mathbf{\Delta_m\%}$ ↑
Single-Task	R18-FPN	64.5	57.4	66.4	68.2	14.8	+ 0.00
	R50-FPN	67.7	61.8	67.2	71.1	14.8	+ 0.00
	R26-DLv3+	64.9	57.1	64.2	71.3	14.9	+ 0.00
	R50-DLv3+	68.3	60.7	65.4	72.7	14.6	+ 0.00
	HRNet-18	60.1	60.7	67.2	69.7	14.6	+ 0.00
ASTMT	R50-FPN	66.8	61.1	66.1	70.9	14.7	- 0.74
	R26-DLv3+	64.6	57.3	64.7	71.0	15.0	- 0.11
	R50-DLv3+	68.9	61.1	65.7	72.4	14.7	- 0.04
PAD-Net	HRNet-18	53.6	59.6	65.8	72.5	15.3	- 5.58
Ours	R18-FPN	65.7	61.1	66.8	73.9	14.6	+ 3.41
Ours	R50-FPN	66.6	63.3	66.6	74.9	14.6	+ 1.32
Ours	HRNet-18	64.3	62.1	68.0	73.4	14.6	+ 3.15

heavily based on DLv3+, we also report results using different backbones. Overall, MTI-Net achieves significantly higher gains over its single-task variants compared to ASTMT (see Δ_m ↑ (ST)). Surprisingly, we find that our model with R18-FPN backbone even outperforms the deeper ASTMT R50-DLv3+ model in terms of multi-tasking performance, despite the fact that the ASTMT single-task models perform better than ours, due to the use of the stronger DLv3+ backbone. Note that we are the first to report consistent multi-task improvements when solving such a diverse task dictionary. Finally, our model also outperforms PAD-Net in terms of multi-tasking performance ($+3.15\%$ vs -5.58%).

Comparison on NYUD

Table 5.5 shows a comparison with the state-of-the-art approaches on NYUD. We leave out methods that rely on extra input modalities, or additional training data. As these methods are built on top of stronger single-scale backbones, we also use the multi-scale HRNet48-v2 backbone here. Again, our model improves w.r.t the single-task models. Furthermore, we perform on par with the state-of-the-art on the depth estimation task, while performing slightly worse on the semantic segmentation task. Figure 5.7 shows some qualitative results.

Table 5.5: Comparison with the state-of-the-art on NYUD.

(a) Results on depth estimation.

Method	**rmse**	**rel**	δ_1	δ_2	δ_3
HCRF [87]	0.821	0.232	0.621	0.886	0.968
DCNF [97]	0.824	0.230	0.614	0.883	0.971
Wang [158]	0.745	0.220	0.605	0.890	0.970
NR forest [134]	0.774	0.187	-	-	-
Xu [166]	0.593	0.125	0.806	0.952	0.986
PAD-Net [165]	0.582	**0.120**	0.817	0.954	0.987
PAP-Net [179]	0.530	0.144	0.815	0.962	0.992
ST - HRNet-48	0.547	0.138	0.828	0.966	0.993
Ours - HRNet-48	**0.529**	0.138	**0.830**	**0.969**	**0.993**

(b) Results on semantic segmentation.

Method	**pixel-acc**	**mean-acc**	**IoU**
FCN [104]	60.0	49.2	29.2
Context [91]	70.0	53.6	40.6
Eigen [41]	65.6	45.1	34.1
B-SegNet [72]	68.0	45.8	32.4
RefineNet-101 [90]	72.8	57.8	44.9
PAD-Net [165]	75.2	62.3	50.2
TRL-ResNet50 [178]	76.2	56.3	46.4
PAP-Net [179]	**76.2**	62.5	**50.4**
ST - HRNet-48	73.4	58.1	45.7
Ours - HRNet-48	75.3	**62.9**	49.0

Resource analysis

We compare our model in terms of computational requirements against PAD-Net and ASTMT. The comparison with PAD-Net is performed on NYUD using the HRNet-18 backbone, while for the comparison with ASTMT on PASCAL we use a ResNet-50 FPN backbone. Table 5.6 reports the results relative to the single-tasking models. On NYUD, MTI-Net reduces the number of FLOPS while improving the performance compared to the single-task models. The reason for the increased amount of parameters is the use of a shallow backbone, and the small number of tasks (i.e. 2). Furthermore, we significantly outperform PAD-Net in terms of FLOPS and multi-task performance. This is due to the fact that PAD-Net performs the multi-modal distillation at a single higher scale (1/4) with $4 \cdot C$ channels, C being the number of backbone channels at a single scale. Instead, we perform most of the computations at smaller scales (1/32,

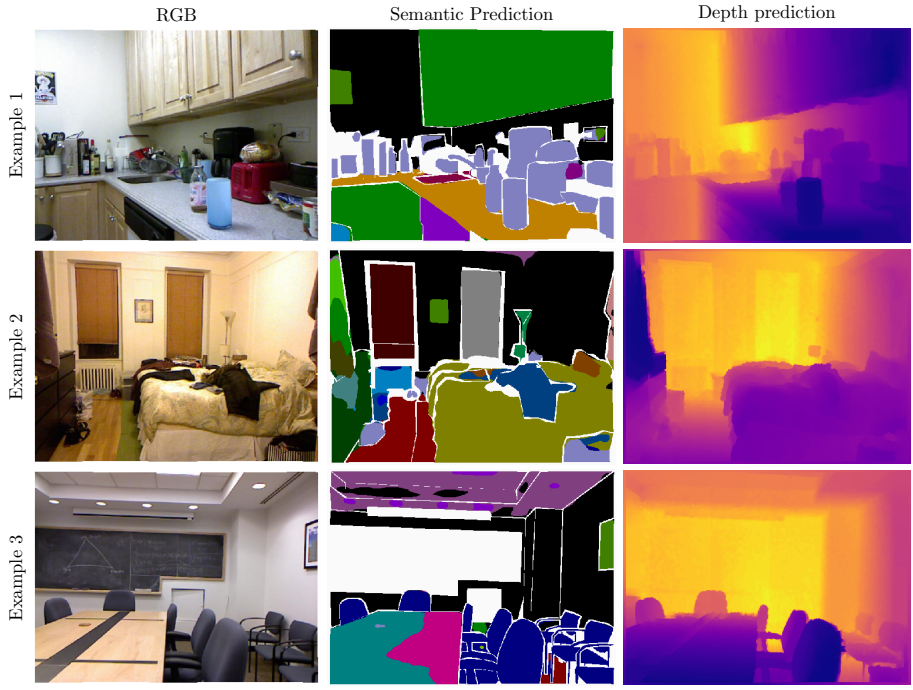


Figure 5.7: Qualitative results on NYUD. We visualize semantic segmentation and depth predictions made by our HRNet-48 model.

1/16, 1/8), while operating on only C channels at the higher scale (1/4). On PASCAL, we significantly improve on all three metrics compared to the single-task models. We also outperform ASTMT in terms of FLOPS and multi-task performance, as the latter has to perform a separate forward pass per task.

5.4 Conclusion

We have shown the importance of modeling task interactions at multiple scales, enabling tasks to maximally benefit each other. We achieved this by introducing dedicated modules on top of an off-the-shelf multi-scale feature extractor, i.e. multi-scale multi-modal distillation, feature propagation across scales, and feature aggregation. Our multi-task model delivers on the full potential of multi-task learning, i.e. smaller memory footprint, reduced number of calculations and better performance. Our experiments show that our multi-task models consistently outperform their single-tasking counterparts by medium to large margins.

Table 5.6: Computational resource analysis (number of parameters and FLOPS).

(a) Results on NYUD (HRNet-18).

Method	**Params** (M)	**FLOPS** (G)	$\mathbf{\Delta_m}\%$
Single Task	8.0	22.0	$+0.00\%$
Multi-Task	-50%	-45%	-1.71%
PAD-Net	-15%	$+204\%$	-0.02%
MTI-Net (Ours)	$+57\%$	-13%	$+6.40\%$

(b) Results on PASCAL (Res-50 FPN).

Method	**Params** (M)	**FLOPS** (G)	$\mathbf{\Delta_m}\%$
Single Task	140	219	$+0.00\%$
Multi-Task	-75.0%	-66%	-4.55%
ASTMT	-51.0%	-1.0%	-0.87%
MTI-Net (Ours)	-35.0%	-19.9%	$+1.36\%$

Limitations MTI-Net was designed with dense prediction tasks in mind. However, its easy to plug-in image-level tasks as well. Since MTI-Net only adds modules in the decoder (after the backbone) to model the task interactions, one could simply include a separate linear decoder for predicting the image-level tasks. After all, image-level tasks are defined at the image-level and do not require a multi-step multi-scale decoding process.

Extensions/ Follow-Up Work We summarize two recent works that proposed extensions to MTI-Net:

- Borse *et al.* [12] proposed a boundary-aware loss term for semantic segmentation using an inverse-transformation network, which learns the degree of parametric transformations between estimated and target boundaries. This plug-in loss can complement the cross-entropy loss in capturing boundary transformations and allows to further improve the results obtained with MTI-Net.
- Bruggemann *et al.* [14] proposed a neural architecture search method to select the refinement units between tasks. The newly generated modules can be used as a drop-in refinement module for any supervised multi-task architecture.

Finally, an interesting direction for future work is combining the design concepts from MTI-Net with recent architectures like transformers.

Chapter 6

Task Balancing Strategies

So far, we focused on designing better architectures for multi-task learning. Another challenge associated with jointly learning multiple tasks is properly weighting the task-specific loss functions. Therefore, in this chapter, we revisit the task balancing techniques from Section 3.2.1. We carefully analyze several existing methods. First, we draw a qualitative comparison listing the most important commonalities and differences between works. Additionally, we perform an experimental evaluation on a variety of benchmarks. We draw several important conclusions: (i) there are various discrepancies between works, (ii) the use of carefully selected fixed weights is competitive with the best performing task balancing methods and (iii) the performance of existing methods is coupled with biases in the training dataset. Based upon these results, we propose a set of heuristics to weigh the task-specific losses. The heuristics are of more practical use and obtain better results across different benchmarks.

The content of this chapter is based upon the following publication:

- *Vandenhende, S., Georgoulis, S., Van Gansbeke, W., Proesmans, M., Dai, D., & Van Gool, L. (2021). Multi-task learning for dense prediction tasks: A survey. IEEE Transactions on Pattern Analysis and Machine Intelligence.*

6.1 Introduction

The multi-task learning optimization objective can be formulated as

$$\mathcal{L}_{MTL} = \sum_i w_i \cdot \mathcal{L}_i \tag{6.1}$$

where w_i and $\mathcal{L}_i$ denote the task-specific weights and loss functions respectively. We need to carefully balance the joint learning of all tasks to avoid a scenario where one or more tasks have a dominant influence in the network weights. This chapter discusses how to achieve this by selecting the task-specific weights w_i in the optimization objective.

There have been several attempts already to automate the selection of the task-specific weights. Note that we provided a detailed description of these works in Chapter 6. We provide a brief recap here. Early work [73] used the homoscedastic uncertainty of each task to weigh the losses. Gradient normalization [27] balances the learning of tasks by dynamically adapting the gradient magnitudes in the network. Liu *et al.* [101] weigh the losses to match the pace at which different tasks are learned. Dynamic task prioritization [51] prioritizes the learning of difficult tasks. Sener *et al.* [139] cast MTL as a multi-objective optimization problem, with the overall objective of finding a Pareto optimal solution.

Section 6.2 compares different task balancing methods from a qualitative point-of-view. Section 6.3 provides an experimental comparison. Based upon the results, we propose a set of heuristics to select the task-specific weights in Equation 6.1.

6.2 Qualitative Comparison

We draw a qualitative comparison between the following methods: uncertainty weighting [73], GradNorm [27], DWA [101], DTP [51] and MGDA [139]. First, we consider whether a method balances the loss magnitudes and/or the pace at which tasks are learned. Second, we show what tasks are prioritized during the training stage, e.g. DTP prioritizes the learning of difficult tasks. Third, methods that require access to the task-specific gradients are indicated. Fourth, we consider whether the task gradients are enforced to be non-conflicting, e.g. MGDA only backpropagates along common directions of the task-specific gradients. Finally, we consider whether the proposed method requires to tune additional hyperparameters, e.g. manually choosing the weights, KPIs, etc.

Table 6.1 shows the results. Our conclusions are summarized below:

- We observe a number of discrepancies between the included methods. For example, uncertainty weighting assigns a higher weight to the 'easy' tasks, while DTP advocates the opposite. The latter can be attributed to the experimental evaluation of the different task balancing strategies that was often done in the literature using different datasets or task dictionaries. We suspect that an appropriate task balancing strategy should be decided for each case individually.

- We also find a few commonalities between the described methods, e.g. both uncertainty weighting, GradNorm and MGDA opted to balance the loss magnitudes as part of their learning strategy.
- Finally, some works (e.g. DWA, DTP) still require careful manual tuning of task-specific hyperparameters, which limits their applicability for dealing with a large number of tasks.

Section 6.3 provides an extensive ablation study under more common datasets or task dictionaries to verify what task balancing strategies are most useful to improve the MTL performance, and under under which circumstances.

6.3 Experiments

6.3.1 Setup

The experiments are performed on three different datasets: CelebA, NYUD and PASCAL. Section 2.2 provides a description of the included datasets. The multi-task models consist of a shared backbone (i.e. ResNet) with task-specific heads. The setup from Chapter 4 is used for CelebA. The backbone is a ResNet-18. We attach a separate linear head for every attribute. The fixed weights scheme simply sums the task-specific losses together, i.e. the task-specific weights are set equal to one. We adopt the setup from Chapter 5 for the experiments on NYUD and PASCAL. A ResNet-18 and ResNet-50 backbone are used for PASCAL and NYUD respectively. The task-specific heads use an ASPP module. On NYUD, we sum the losses together, meaning the fixed weight scheme overlaps with the uniform weights. Differently, on PASCAL, we obtain the fixed weights from a grid search: $w_{seg} = 1.0$, $w_{parts} = 2.0$, $w_{sal} = 5.0$, $w_{edge} = 50.0$ and $w_{normals} = 10.0$.

Table 6.1: A qualitative comparison between task balancing techniques. First, we consider whether a method balances the loss magnitudes (**Balance Magnitudes**) and/or the pace at which tasks are learned (**Balance Learning**). Second, we show what tasks are prioritized during the training stage (**Prioritize**). Third, we show whether the method requires access to the task-specific gradients (**Grads Required**). Fourth, we consider whether the tasks gradients are enforced to be non-conflicting (**Non-Competing Grads**). Finally, we consider whether the proposed method requires additional tuning, e.g. manually choosing the weights, KPIs, etc. (**No Extra Tuning**). For a detailed description of each technique see Section 3.2.

Method	**Balance Magnitudes**	**Balance Learning**	**Prioritize**	**Grads Required**	**Non-Competing Grads**	**No Extra Tuning**	**Motivation**
Uncertainty [73]	✓		Low Noise			✓	Homoscedastic uncertainty
GradNorm [27]	✓	✓		✓		✓	Balance learning and magnitudes
DWA [101]		✓					Balance learning
DTP [51]			Difficult				Prioritize difficult tasks
MGDA [139]	✓	✓		✓	✓	✓	Pareto Optimum

6.3.2 Results

CelebA

Table 6.2a reports the results for CelebA. The multi-task model trained with fixed uniform weights outperforms the single-task networks (91.91% vs 91.52%). We argue that the additional supervisory signals from other attributes can act as a regularizer for the backbone features. Surprisingly, none of the proposed task balancing strategies improves over the model trained with fixed uniform weights on CelebA, although all methods are very close. We conclude that the uniform weighting scheme is already close to optimal in case of multi-attribute learning, where all individual losses are binary cross-entropy losses. Note that, a similar experiment was performed in [139], but different conclusions were reported. First, they found that the multi-task model trained with fixed weights performs worse when compared against the single-task networks. Second, the authors observed that the loss balancing strategies improve over the uniform weights. In this manuscript, we find that with sufficient and comparable amounts of hyperparameter tuning both conclusions no longer stand.

NYUD

Table 6.2b shows the results when training the MTL baseline with a ResNet-50 backbone using different task balancing strategies. On NYUD, optimizing the loss weights with a grid search procedure resulted in a uniform loss weighting scheme. Therefore, in this case, the use of fixed uniform weights and the use of fixed weights from a grid search overlap.

The MTL baseline with fixed weights improves over the single-task networks ($+0.41\%$ for Δ_{MTL}). GradNorm can further boost the performance by adjusting the task-specific weights in the loss during training ($+1.45\%$). We conclude that uniform weights are suboptimal for training a multi-task model when the tasks employ different loss functions, although the differences are limited.

Similar to GradNorm, DWA tries to balance the pace at which tasks are learned, but does not equalize the gradient magnitudes. From Table 6.2b, we conclude that the latter is important as well (-0.28% with DWA vs $+1.45\%$ with GradNorm). Uncertainty weighting results in reduced performance compared to GradNorm (-0.23% vs 1.45%). Uncertainty weighting assigns a smaller weight to noisy or difficult tasks. Since the annotations on NYUD were not distilled, the noise levels are rather small. When we have access to clean ground-truth annotations, it seems better to balance the learning of tasks rather than lower the weights for the difficult tasks. Further, MGDA reports lower performance compared to GradNorm ($+0.02\%$ vs $+1.45\%$). MGDA only updates the weights along directions that are common among all task-specific gradients. The results

Table 6.2: Comparison of different tasks balancing techniques. The setup is described in Section 6.3.1.

(a) Results on CelebA for 40 person attribute classification tasks.

Method	**Accuracy** (%) ↑
Single-task	91.52
Fixed	**91.91**
Uncertainty	91.62
GradNorm	91.76
DWA	91.84
MGDA	91.30

(b) Results on NYUD.

Method	**Seg.** (IoU) ↑	**Depth** (rmse) ↓	Δ_{MTL} (%)↑
Single-Task	43.9	0.585	+ 0.00
Fixed	**44.4**	0.587	+ 0.41
Uncertainty	44.0	0.590	- 0.23
DWA	44.1	0.591	- 0.28
GradNorm	44.2	**0.581**	**+ 1.45**
MGDA	43.2	0.576	+ 0.02

(c) Results on PASCAL.

Method	**Seg.** (IoU) ↑	**H. Parts** (IoU) ↑	**Norm.** (mErr) ↓	**Sal.** (IoU) ↑	**Edge** (odsF) ↑	Δ_{MTL} (%) ↑
Single-Task	**66.2**	**59.9**	**13.9**	**66.3**	68.8	**+ 0.00**
Fixed (Grid S.)	63.8	58.6	14.9	65.1	**69.2**	- 2.86
Uncertainty	65.4	59.2	16.5	65.6	68.6	- 4.60
DWA	63.4	58.9	14.9	65.1	69.1	- 2.94
GradNorm	64.7	59.0	15.4	64.5	67.0	- 3.97
MGDA	64.9	57.9	15.6	62.5	61.4	-6.81

suggest that it is better to allow some competition between the task-specific gradients in the shared layers, as this could help to avoid local minima.

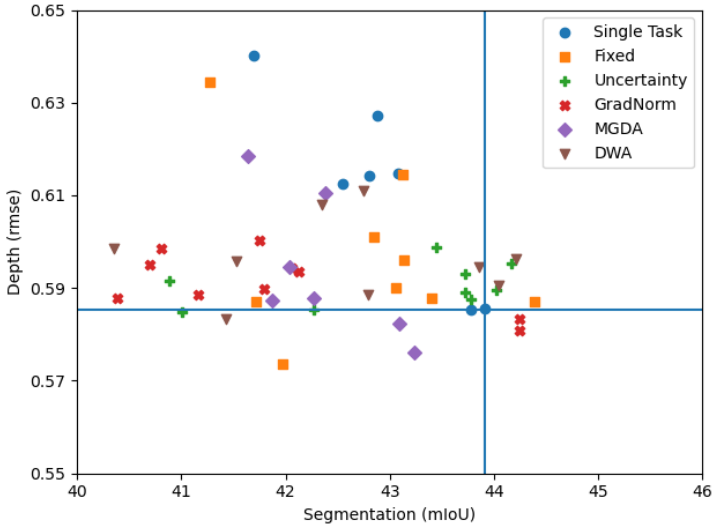
Finally, we conclude that the use of fixed loss weights optimized with a grid search still outperforms several existing task balancing methods. In particular, the solution obtained with fixed uniform weights outperforms the models trained with uncertainty weighting, MGDA and DWA on both the semantic segmentation and depth estimation tasks. Perhaps not surprisingly, we found that the uniform weights resulted in task-specific losses of similar magnitudes. In other words, it seems sufficient to make sure that the losses are of similar magnitude to obtain good results.

PASCAL

The task balancing techniques are compared on PASCAL in Table 6.2c. We find that a grid-search on the weight space works better than the use of automated task balancing procedures. A similar observation was made by [110]. We hypothesize that this is due to the imbalance between the optimal parameters, e.g. the weight for the edge detection loss is 50 times higher than for the semantic segmentation loss. Also, we observe that the grid-search weights result in task-specific losses of similar magnitudes. This again hints that its useful to ensure that the losses are of similar magnitude.

Uncertainty weighting reports the highest performance losses on the distilled tasks, i.e. normals and saliency. This is because uncertainty weighting assigns a smaller weight to tasks with higher homoscedastic uncertainty. Differently, MGDA fails to properly learn the edge detection task. This is another indication (cf. NYUD results) that avoiding competing gradients by only back propagating along a common direction in the shared layers does not necessarily improve performance. This is particularly the case for the edge detection task on PASCAL because of two reasons. First, the edge detection loss has a much smaller magnitude compared to the other tasks. Second, when the edge annotations are converted into segments they provide an over-segmentation of the image. As a consequence, the loss gradient of the edge detection task often conflicts with other tasks since they have a smoother gradient. Because of this, MGDA prefers to mask out the gradient from the edge detection task by assigning it a smaller weight. Finally, GradNorm does not report higher performance when compared against uniform weighting. We hypothesize that this is due to the re-normalization of the loss weights during the optimization. The latter does not work well when the optimal loss magnitudes are very imbalanced.

Figure 6.1: Performance profile of task balancing strategies for the semantic segmentation and depth estimation tasks on NYUD. We show the results obtained with different hyperparameter settings in the same color. Bottom-right is better.



Performance Profiles

We use the performance profile from Figure 6.1 to analyze the hyperparameter sensitivity of the task balancing methods on NYUD. GradNorm is the only technique that outperforms training a separate model for each task. However, when less effort is spent to tune the hyperparameters of the single-task models, all task balancing techniques result in improved performance. This observation explains the increased performance relative to the single-task case reported by prior works [73, 27, 139]. We stress the importance of carefully training the single-task baselines.

Second, the MTL models seem more robust to hyperparameter changes compared to the single-task networks. In particular, the performance drops slower when using less optimal hyperparameter settings in the MTL case.

Finally, fixed loss weighting and uncertainty weighting are most robust against hyperparameter changes. These techniques report high performance for a large number of hyperparameter settings. For example, out of the top-20% best models, 40% of those were trained with uncertainty weighting.

6.3.3 Discussion

To our surprise, we found that grid-search is competitive or better compared to existing task balancing techniques. Unfortunately, performing a grid search becomes intractable, if not infeasible, when considering large task dictionaries. However, there are several tricks that practitioners can apply to circumvent this problem. In particular, we recommend the adoption of a set of heuristics to obtain an effective task weighting in an efficient manner. We observed that the task-specific weights are competitive with the ones from the grid-search procedure.

- **Heuristic 1: Related tasks using the same loss functions can use the same weights.** For example, on CelebA, the person attribute classification tasks are strongly related and use the same loss functions. In this case, using uniform, fixed weights gives good results (see Table 6.2a).
- **Heuristic 2: If the tasks use different task-specific losses, then adjust the task-specific weights such that all losses are of similar magnitude.** This can be realized by tracking the task-specific losses over a few iterations for every task. The task-specific weights can be set based upon the averaged loss values. For example, assume two tasks with the average loss value of the first task being 100 times larger compared to the average loss value of task 2. In this case, the weights are set as $w_1 = 1$ and $w_2 = 100$. This trick applies to PASCAL and NYUD (see Tables 6.2b-6.2c). We verified that the obtained weights are similar to the ones from the grid search and the differences in performance are negligible.
- **Heuristic 3: Increase the task-specific weights for tasks that are more important.** This ensures more effort is spend to optimize the tasks of interest.
- **Heuristic 4: Increase the task-specific weights for tasks that are more difficult.** For example, the task-specific weights could be updated at regular intervals during training following the procedure outlined for Heuristic 2. This heuristic is based upon Dynamic Task Prioritization [51].
- **Heuristic 5: When multiple datasets with different task dictionaries are used, compose the batches such that they show more examples for the more challenging tasks.** Bell *et al.* [10] show how this can be realized. Assume n datasets. The design space is explored by performing two-pair dataset ablations to understand the relative balance of the datasets. For example, if for some task the accuracy is insensitive to the batch size, then we can reduce that dataset's contribution to each batch, leaving more slots for the challenging tasks.

Further, a number of techniques performed worse than anticipated. Gong *et al.* [48] obtained similar results with ours, albeit only for a few loss balancing strategies.

Also, Maninis *et al.* [110] found that performing a grid search for the weights could outperform a state-of-the-art loss balancing scheme. These observations seem particularly true for larger datasets with more diverse task dictionaries like PASCAL (see Table 6.2c). We argue that new methods should also evaluate their approach under more challenging conditions. Finally, existing task balancing methods can still be useful to train MTL networks. For example, when dealing with noisy annotations, uncertainty weighting can help to automatically readjust the weights on the noisy tasks.

6.4 Conclusion

We analyzed multiple task balancing strategies and isolated the elements that are most effective for balancing task learning, e.g. down-weighing noisy tasks, balancing loss magnitudes, etc. Yet, many optimization aspects still remain poorly understood. For example, opposed to recent works, our analysis indicates that avoiding gradient competition between tasks can hurt performance. Furthermore, our study revealed that some task balancing strategies still suffer from shortcomings and highlighted several discrepancies between existing methods. Finally, we proposed a set of heuristics based upon our experimental results. These aim to make it easier for practitioners to train multi-task learning models in an effective way.

Limitations The optimization strategies have been studied using the multi-task learning baseline model. It would be interesting to see if our conclusions translate to other, possibly more sophisticated models. Further, we studied the optimization aspect from an experimental point-of-view. For future work, it would be useful to develop a theoretical framework to examine these methods.

Chapter 7

Self-Supervised Learning

The experiments in this thesis were performed using supervised pretraining on ImageNet. This setup builds upon the fact that the weights of convolutional neural networks transfer well between tasks. As a result, pretraining on a large dataset like ImageNet allows to perform well on tasks with only a modest amount of labelled data (e.g. segmentation on NYUD or PASCAL). Pretraining is a powerful technique that is widely known and applies to both single- and multi-task learning.

Unfortunately, supervised pretraining relies on an annotated pretraining dataset, which limits it's applicability. To resolve this problem, researchers have begun to explore self-supervised learning methods. The latter group of methods learn visual representations by solving *pretext tasks* for which the learning signal comes for free. In this chapter, we aim to explore the use of self-supervised pretraining to improve models for scene understanding.

Recently, self-supervised learning has outperformed supervised pretraining on many downstream tasks like segmentation and object detection. However, current methods are still primarily applied to curated datasets like ImageNet. In this chapter, we first study how biases in the dataset affect existing methods. Our results show that an approach like MoCo [56] works surprisingly well across: (i) object- versus scene-centric, (ii) uniform versus long-tailed and (iii) general versus domain-specific datasets. Second, given the generality of the approach, we try to realize further gains with minor modifications. We show that learning additional invariances - through the use of multi-scale cropping, stronger augmentations and nearest neighbors - improves the representations. Next, we observe that MoCo learns spatially structured representations when trained with a multi-crop strategy. The representations can be used for semantic segment retrieval and video instance segmentation without finetuning. Moreover, the results are on par with specialized models. Finally, we apply contrastive self-supervised

pretraining to improve a multi-task learning model for indoor scene understanding.

The content of this chapter is based upon the following preprint[1]:

- Van Gansbeke, W.*, **Vandenhende, S.***, Georgoulis, S., & Van Gool, L. (2021). Revisiting Contrastive Methods for Unsupervised Learning of Visual Representations. NeurIPS.

7.1 Introduction

Self-supervised learning (SSL) [71] aims to learn powerful representations without relying on human annotations. The representations can be used for various purposes, including transfer learning [56], clustering [6, 148, 150] or semi-supervised learning [22]. Recent self-supervised methods [21, 56, 16, 50] learn visual representations by imposing invariances to various data transformations. A popular way of formulating this idea is through the instance discrimination task [162] - which treats each image as a separate class. Augmentations of the same image are considered as positive examples of the class, while other images serve as negatives. To handle the large number of instance classes, the task is expressed as a non-parametric classification problem using the contrastive loss [54, 120].

Despite the recent progress, most methods still train on images from ImageNet [33]. This dataset has specific properties: (1) the images depict a single object in the center of the image, (2) the classes follow a uniform distribution and (3) the images have discriminative visual features. To deploy self-supervised learning into the wild, we need to quantify the dependence on these properties. Therefore, in this chapter, we first study the influence of dataset *biases* on the representations. We take a utilitarian view and transfer different representations to a variety of downstream tasks.

Our results indicate that an approach like MoCo [56] works well for both object- and scene-centric datasets. We delve deeper to understand these results. A key component is the augmentation strategy which involves random cropping. For an object-centric dataset like ImageNet, two crops from the same image will show a portion of the same object and no other objects. However, when multiple objects are present, a positive pair of non-overlapping crops could lead us to wrongfully match the feature representations of different objects. This line of thought led recent studies [128, 138] to believe that contrastive SSL benefits from object-centric data.

So how can contrastive methods learn useful representations when applied to more complex, scene-centric images? We propose a hypothesis that is two-fold. First, the default parameterization of the augmentation strategy avoids non-overlapping views.

[1] * denotes equal contribution.

As a result, positive pairs will share information, which means that we can match their representations. Second, when applying more aggressive cropping, we only observe a small drop in the transfer learning performance. Since patches within the same image are strongly correlated, maximizing the agreement between non-overlapping views still provides a useful learning signal. We conclude that, in visual pretraining, combining the instance discrimination task with random cropping is universally applicable.

The common theme of recent advances is to learn representations that are invariant to different transformations. Starting from this principle, we try to improve the results obtained with an existing framework [56]. More specifically, we investigate three ways of generating a more diverse set of positive pairs. First, the multi-crop transform from [16] is revisited. Second, we examine the use of a stronger augmentation policy. Third, we leverage nearest neighbors mined online during training as positive views. The latter imposes invariances which are difficult to learn using handcrafted image transformations. The proposed implementation requires only a few lines of code and provides a simple, yet effective alternative to clustering based methods [16, 88]. Each of the proposed additions is found to boost the performance of the representations under the transfer learning setup.

The multi-crop transform realizes significant gains. We probe into what the network learns to explain the improvements. The multi-crop transform maximizes the agreement between smaller (local) crops and a larger (global) view of the image. This forces the model to learn a more spatially structured representation of the scene. As a result, the representations can be directly used to solve several dense prediction tasks without any finetuning. In particular, we observe that the representations already model class semantics and dense correspondences. Furthermore, the representations are competitive with specialized methods [68, 177]. In conclusion, the multi-crop setup provides a viable alternative to learn dense representations without relying on video data [68, 83] or handcrafted priors [67, 177, 150].

In summary, the overall goal of this chapter is to learn more effective representations through contrastive self-supervised learning without relying too much on specific dataset biases. We further illustrate the usefulness of self-supervised learning for an application in multi-task learning. The remainder of this chapter is structured as follows. Section 7.2 introduces the framework. Section 7.3 studies whether the standard SimCLR augmentations transfer across different datasets. This question is answered positively. Based upon this result, Section 7.4 then studies the use of additional invariances to further improve the learned representations. Finally, Section 7.6 applies our pretrained weights to improve a multi-task learning model for indoor scene understanding. We hope this chapter will provide useful insights to other researchers. .

7.2 Framework

We briefly introduce the contrastive learning framework. The idea is to generate feature representations that maximize the agreement between similar (*positive*) images and minimize the agreement between dissimilar (*negative*) images. Let x be an image. Assume that a set of positives for x can be acquired, denoted by $\mathcal{X}^+$. Similarly, a set of negatives $\mathcal{X}^-$ is defined. We learn an embedding function f that maps each sample on a normalized hypersphere. The contrastive loss [54, 120] takes the following form

$$\mathcal{L}_{\text{contrastive}} = -\sum_{x^+ \in \mathcal{X}^+} \log \frac{\exp\left[f(x)^T \cdot f(x^+)/\tau\right]}{\exp\left[f(x)^T \cdot f(x^+)/\tau\right] + \sum_{x^- \in \mathcal{X}^-} \exp\left[f(x)^T \cdot f(x^-)/\tau\right]} \tag{7.1}$$

where τ is a temperature hyperparameter. We will further refer to the image x as the *anchor*.

SSL methods obtain positives and negatives by treating each image as a separate class [162]. More specifically, augmented views of the same image are considered as positives, while other images are used as negatives. The data augmentation strategy is an important design choice as it determines the invariances that will be learned. Today, most works rely on a similar set of augmentations that consists of (1) cropping, (2) color distortions, (3) horizontal flips and (4) Gaussian blur.

In this manuscript, we build upon MoCo [56] - a widely known and competitive framework. However, our findings are expected to apply to other related methods (e.g. SimCLR [21]) as well. The embedding function f with parameters θ_f consists of a backbone g (e.g. ResNet [59]) and a projection MLP head h. The contrastive loss is applied after the projection head h. MoCo uses a queue and a moving-averaged encoder f' to keep a large and consistent set of negative samples. The parameters $\theta_{f'}$ of f' are updated as: $\theta_{f'} = m\theta_{f'} + (1-m)\theta_f$ with m a momentum hyperparameter. The momentum-averaged encoder f' takes as input the anchor x, while the encoder f is responsible for the positives $\mathcal{X}^+$. The queue maintains the encoded anchors as negatives. Algorithm 2 provides the pseudocode of MoCo. We refer to [56] for more details.

7.3 Contrastive Learning in the Wild

Most contrastive self-supervised methods train on unlabeled images from ImageNet [33]. This is a curated dataset with unique characteristics. First, the images are *object-centric*, i.e. they depict only a single object in the center of the image. This differs from other datasets [94, 82] which contain more complex scenes with several objects. Second, the underlying classes are *uniformly* distributed. Third, the images

Algorithm 2 Pseudocode of MoCo [56] in a PyTorch-like style.

```
# f, f_m: the model f and momentum updated model f_m
# f consists of the backbone g and projection head h
# queue: dictionary as a queue of K keys (C x K)
# m: momentum
# t: temperature

f_m.params = f.params # initialize momentum updated backbone

for batch in loader: # load a minibatch with B samples

   # randomly augment batch
   x_anchors = aug(batch)
   x_positives = aug(batch)

   # forward pass
   anchors, positives = f_m(x_anchors), f(x_positives) # B x C
   anchors = anchors.detach() # no gradient

   # compute logits
   l_pos = bmm(positives.view(B,1,C), anchors.view(B,C,1)) # B x 1
   l_neg = mm(positives, queue.view(C,K)) # B x K
   logits = cat([l_pos, l_neg], dim=1) # B x (K+1)

   # loss: (1) sharpen with temperature t, (2) apply cross-entropy loss
   loss = CE(logits/t, zeros(B))

   # SGD update
   loss.backward()
   update(f.params)

   # momentum update
   f_m.params = m * f_m.params + (1-m) * f.params

   # update dictionaries
   enqueue_dequeue(queue, anchors)
```

`bmm`: batch matrix mult.; `mm`: matrix mult.; `cat`: concatenate; `CE`: cross-entropy loss

have *discriminative* visual features. For example, ImageNet covers various bird species which can be distinguished by a few key features. In contrast, domain-specific datasets (e.g. BDD100K [172]) contain less discriminative scenery showing the same objects like cars, pedestrians, etc. In this section, we study the influence of dataset biases for contrastive self-supervised methods.

7.3.1 Setup

We train MoCo-v2 [23] on a variety of datasets. Table 7.1 shows an overview. Figure 7.1 shows a representative sample from each dataset. The representations are evaluated on six downstream tasks: linear classification, semantic segmentation, object detection, video instance segmentation and depth estimation. We adopt the following target datasets for linear classification: CIFAR10 [80], Food-101 [78], Pets [122], Places365 [184], Stanford Cars [78], SUN397 [163] and VOC 2007 [45]. The semantic segmentation task is evaluated on Cityscapes [29], PASCAL VOC [45] and NYUD [140]. We use PASCAL VOC [45] for object detection. The DAVIS-2017

Table 7.1: Overview of the pretraining datasets. We sample a uniform and long-tailed (LT) subset of 118K images from ImageNet. On OpenImages, we sample a random subset of 118K images. The complete train splits are used for COCO and BDD100K.

Pretrain Data	#Imgs	#Obj/Img	Uniform	Discriminative
ImageNet-118K [33]	118 K	1.7	✓	✓
ImageNet-118K-LT [33]	118 K	1.7	✗	✓
COCO [94]	118 K	7.3	✗	✓
OpenImages-118K [82]	118 K	8.4	✗	✓
BDD100K [172]	90 K	-	✗	✗



Figure 7.1: Example images from the pretraining datasets.

benchmark [127] is used for video instance segmentation. Finally, depth estimation is performed on NYUD [140]. The model, i.e. a ResNet-50 backbone, is pretrained for 400 epochs using batches of size 256. The initial learning rate is set to 0.3 and decayed using a cosine schedule. We use the default values for the temperature ($\tau = 0.2$) and momentum ($m = 0.999$) hyperparameters.

Table 7.2: Comparison of linear classification models trained on top of frozen features (400 epochs pretraining).

Pretrain Data	CIFAR10	Cars	Food	Pets	Places	SUN	VOC
IN-118K	83.1	35.9	62.2	68.9	45.0	50.0	75.8
COCO	77.4 (−5.7)	33.9 (−2.0)	62.0 (−0.2)	62.6 (−6.3)	47.3 (+2.3)	53.6 (+3.6)	80.9 (+5.1)
OI-118K	74.0 (−9.1)	32.2 (−3.7)	58.4 (−3.8)	59.3 (−9.6)	46.6 (+1.6)	52.3 (+2.3)	75.9 (+0.1)
IN-118K-LT	84.0 (+0.9)	37.2 (+1.3)	62.3 (+0.1)	72.4 (+3.5)	45.7 (+0.7)	50.7 (+0.7)	77.6 (+1.8)

Table 7.3: Comparison of different representations under the transfer learning setup (400 epochs pretraining).

	Semantic seg. (mIoU)			**Detection** (AP)	**Vid. seg.** ($\mathcal{J}\&\mathcal{F}$)	**Depth** (rmse)
Pretrain Data	**VOC**	**Cityscapes**	**NYUD**	**VOC**	**DAVIS**	**NYUD**
IN-118K	68.9	70.1	37.7	53.0	63.5	0.625
COCO	69.1 (+0.2)	70.3 (+0.2)	39.3 (+1.6)	53.0 (+0.0)	65.1 (+1.6)	0.612 (−0.013)
OI-118K	67.9 (−1.0)	70.9 (+0.8)	38.4 (+0.7)	53.1 (+0.1)	64.8 (+1.3)	0.609 (−0.016)
IN-118K-LT	69.2 (+0.3)	70.2 (+0.1)	38.5 (+0.8)	53.1 (+0.1)	63.7 (+0.2)	0.610 (−0.015)
BDD100K	-	70.1 (+0.0)	-	-	-	-

7.3.2 Object-centric Versus Scene-centric

We compare the representations trained on an object-centric dataset - i.e. ImageNet (IN-118K) - against the ones obtained from two scene-centric datasets - i.e. COCO and OpenImages (OI-118K). Tables 7.2-7.3 show the results under the linear classification and transfer learning setup.

Results. The linear classification model yields better results on CIFAR10, Cars, Food and Pets when pretraining the representations on ImageNet. Differently, when tackling the classification task on Places, SUN and VOC, the representations from COCO and OpenImages are better suited. The first group of target benchmarks contains images centered around a single object, while the second group contains scene-centric images with multiple objects. We conclude that, for linear classification, the pretraining dataset should match the target dataset in terms of being object- or scene-centric.

Next, we consider finetuning. Perhaps surprisingly, we do not observe any significant disadvantages when using more complex images from COCO or OpenImages for pretraining. In particular, for the reported tasks, the COCO pretrained model even outperforms its ImageNet counterpart. We made a similar observation when increasing the size of the pretraining dataset (see [149]).

Discussion. In contrast to prior belief [128, 138], our results indicate that contrastive self-supervised methods do not suffer from pretraining on scene-centric datasets. How can we explain this inconsistency with earlier studies? First, the experimental setup in [128] only considered the linear evaluation protocol for an object-centric dataset (i.e. PASCAL cropped boxes). This analysis [128] does not show us the full picture. Second, the authors conclude that existing methods suffer from using scene-centric datasets due to the augmentation strategy, which involves random cropping. They argue that aggressive cropping could yield non-overlapping views which contain different objects. In this case, maximizing the feature similarity would be detrimental for object recognition tasks. However, the default cropping strategy barely yields non-overlapping views[2]. This is verified by plotting the intersection over union (IoU) between pairs of crops (see Figure 7.2). We conclude that the used example of non-overlapping crops [128, 138] seldom occurs.

The above observation motivates us to reconsider the importance of using overlapping views. We pretrain on COCO while forcing the IoU between views to be smaller than a predefined threshold. Figures 7.3-7.4 show the transfer performance and training curves for different values of the threshold. The optimization objective is harder to satisfy when applying more aggressive cropping (i.e. the training loss increases when lowering

[2] We use the `RandomResizedCrop` in PyTorch with scaling $s = (0.2, 1.0)$ and output size 224×224.

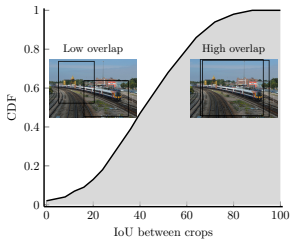


Figure 7.2: IoU between random resized crops for existing frameworks.

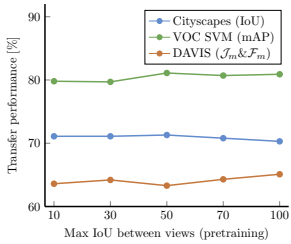


Figure 7.3: Transfer results when thresholding the IoU between crops.

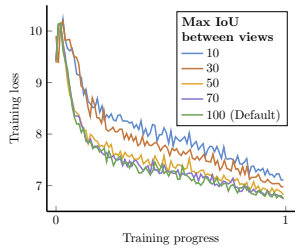


Figure 7.4: Training curves when thresholding the IoU between crops.

the IoU). However, Figure 7.3 shows that the transfer performance remains stable. Patches within the same image were observed at the same point in time and space, which means that they will share information like the camera viewpoint, color, shape, etc. As a result, the learning signal is still meaningful, even when less overlapping crops are used as positives.

7.3.3 Uniform Versus Long-tailed

Next, we study whether MoCo benefits from using a uniform (IN-118K) versus long-tailed (IN-118K-LT) dataset. We adopt the sampling strategy from [103] to construct a long-tailed version of ImageNet. The classes follow the Pareto distribution with power value $\alpha = 6$. Tables 7.2-7.3 indicate that MoCo is robust to changes in the class distribution of the dataset. In particular, the IN-118K-LT model performs on par or better compared to its IN-118K counterpart across all tasks. We conclude that it is not essential to use a uniformly distributed dataset for pretraining.

7.3.4 Domain-Specific Datasets

Images from ImageNet have discriminative visual features, e.g. the included bird species can be recognized from their beak or plumage. In this case, the model could achieve high performance on the instance discrimination task by solely focusing on the most discriminative feature in the image. Differently, in urban scene understanding, we deal with more monotonous scenery - i.e. all images contain parked cars, lane markings, pedestrians, etc. We try to measure the impact of using less discriminative images for representation learning.

We pretrain on BDD100K - a dataset for urban scene understanding. Table 7.3 compares the representations for the semantic segmentation task on Cityscapes when pretraining on BDD100K versus IN-118K. The BDD100K model performs on par with the IN-118K pretrained baseline (BDD100K: $+0.00\%$). This shows that MoCo can be applied to domain-specific data as well. Two models trained on very different types of data perform equally well. A recent study [181] showed that it is mostly the low- and mid-level visual features that are retained under the transfer learning setup. This effect explains how two different representations produce similar results. Finally, we expect that further gains can be achieved by finetuning the augmentation strategy using domain knowledge.

7.4 Learning Invariances

In Section 7.3, we showed that MoCo works well for a large variety of pretraining datasets: scene-centric, long-tailed and domain-specific images. This result shows that there is no obvious need to use a more advanced pretext task [138] or use an alternative source of data like video [128] to improve the representations. Therefore, instead of developing a new framework, we try to realize further gains while sticking with the approach from MoCo. We realize this objective by learning additional invariances. We study three such mechanisms, i.e. multi-scale constrained cropping, stronger augmentations and the use of nearest neighbors. We concentrate on the implementation details and uncover several interesting qualities of the learned representations. Table **??** shows an overview of the results.

7.4.1 Multi-Scale Constrained Cropping

The employed cropping strategy proves crucial to obtain powerful visual representations. So far, we used the default two-crop transform, which samples two image patches to produce an anchor and a positive. The analysis in Section 7.3 showed that this transformation yields highly-overlapping views. As a result, the model can match the anchor with its positive by attending only to the most discriminative image component, while ignoring other possibly relevant regions. This observation leads us to revisit the `multi-crop` augmentation strategy from [16].

Table 7.4: **Ablation of different components.** Models are pretrained for 200 epochs on COCO using the settings from Section 7.3. We indicate the differences with MoCo. The best model is trained with additional invariances.

	Setup					Semantic seg. (mIoU)			Classification (mAP / Acc. / Acc.)		
Method	MC	CC	m↓	A^+	NN	VOC	Cityscapes	NYUD	VOC	ImageNet	Places
Rand. init.	-	-	-	-	-	39.2	65.0	24.4	-	-	-
MoCo	✗	✗	✗	✗	✗	66.2	70.3	38.2	76.1	49.3	45.1
Sec. 7.4.1	✓	✗	✗	✗	✗	69.9 (+3.7)	70.9 (+0.6)	39.4 (+1.2)	81.3 (+5.2)	53.4 (+4.1)	47.7 (+2.6)
	✓	✓	✗	✗	✗	70.2 (+4.0)	70.9 (+0.6)	39.5 (+1.3)	82.1 (+6.0)	54.0 (+4.7)	47.9 (+2.8)
	✓	✓	✓	✗	✗	70.9 (+4.7)	71.3 (+1.0)	39.9 (+1.7)	82.8 (+6.7)	54.8 (+5.5)	48.1 (+3.0)
Sec. 7.4.2	✓	✓	✓	✓	✗	71.4 (+5.2)	72.0 (+1.7)	40.0 (+1.8)	83.7 (+7.6)	55.5 (+6.2)	48.2 (+3.1)
Sec. 7.4.3	✓	✓	✓	✓	✓	71.9 **(+5.7)**	72.2 **(+1.9)**	40.9 **(+2.7)**	85.1 **(+9.0)**	55.9 **(+6.6)**	48.5 **(+3.4)**

MC: Multi-crop, CC: Constrained multi-crop, m↓: Lower momentum, A^+: Stronger augmentations, NN: Nearest neighbors



Figure 7.5: The constrained multi-crop. Smaller crops are forced to lie within the anchor image. Invalid/valid crops are colored in red/green.

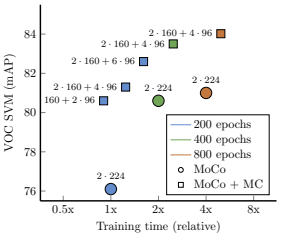


Figure 7.6: Transfer performance versus time for the multi-crop (MC) model.

Setup The `multi-crop` transform samples N additional positive views for each anchor. Remember, the views are generated by randomly applying resized cropping, color distortions, horizontal flipping and Gaussian blur. We adjust the scaling parameters and resolution of the random crop transform to limit the computational overhead. The two default crops (i.e. the anchor and a positive) contain between 20% and 100% of the original image area, and are resized to 160×160 pixels. Differently, the N additional views contain between 5% and 14% of the original image area, and are resized to 96×96 pixels. We maximize the agreement between all $N + 1$ positives and the anchor. Note that the N smaller, more aggressively scaled crops are not used as anchors, i.e. they are not shown to the momentum-updated encoder and are not used as negatives in the memory bank.

Further, we consider two additional modifications. First, we enforce the smaller crops to overlap with the anchor image. In this way, all positives can be matched with their anchor as there is shared information. The new transformation is referred to as `constrained multi-cropping`. Figure 7.5 illustrates the concept. Second, since increasing the number of views facilitates faster training, we reduce the momentum hyperparameter m from 0.999 to 0.995. We pretrain for 200 epochs on COCO.

Results Table 7.4 benchmarks the modifications. Both the `multi-crop` (MC) and `constrained multi-crop` (CC) significantly improve the transfer performance. Reducing the momentum (m↓) yields further gains. Figure 7.6 plots the training time versus performance for the `multi-crop` model. The `multi-crop` improves the computational efficiency w.r.t. the two-crop transform.

Discussion We investigate what renders the `multi-crop` and `constrained multi-crop` effective. The network matches the representations of the large anchor

image with the small crops. Since the small crops show random subparts of the anchor image, the network is forced to encode as much information as possible, such that all image regions can be retrieved from the representation of the scene. As a result, the representations will be informative of the spatial layout and different objects in the image - rather than attend only to the most discriminative image component. The latter is verified by visualizing the class activation maps (CAMs) [183] of different representations (see Figure 7.7). The CAMs of the `multi-crop` model segment the complete object, while the `two-crop` model only looks at a few discriminative components.

Further, the representations are evaluated on two dense prediction tasks without any finetuning. We consider the tasks of video instance segmentation on DAVIS 2017 [127] and semantic segment retrieval on VOC [45]. The `multi-crop` model is found superior at modeling dense correspondences for the video instance segmentation task in Table 7.5 (MoCo vs. MoCo + MC). Similarly, Table 7.6 shows that the `multi-crop` model achieves higher performance on the semantic segment retrieval task compared to its two-crop counterpart. This indicates that the pixel embeddings are better disentangled according to the semantic classes. Finally, the MoCo `multi-crop` model is competitive with other methods that were specifically designed for the tasks. We conclude that the `multi-crop` setup provides a viable alternative to learn dense representations without supervision. Moreover, this setup does not rely on video [83, 68] or handcrafted priors [67, 177, 150].

7.4.2 Stronger Augmentations

As mentioned before, the image crops are augmented by random color distortions, horizontal flips and Gaussian blur. Can we do better by using stronger augmentations? We investigate the use of AutoAugment [30] - an advanced augmentation policy obtained using supervised learning on ImageNet. We consider three possible strategies to augment the positives: (1) standard augmentations, (2) AutoAugment and (3) randomly applying either (1) or (2).

Table 7.7 compares the representations under the linear evaluation protocol on PASCAL VOC. Replacing the standard augmentations with AutoAugment degrades the performance (from 82.8% to 79.9%). However, randomly applying either the standard augmentations or AutoAugment does improve the result (from 82.8% to 83.7%). Chen *et al.* [21] showed that contrastive SSL benefits from using strong color augmentations as two random crops from the same image will share a similar color distribution. AutoAugment applies fewer color distortions, resulting in lower performance. This is compensated when combining both augmentation strategies. Finally, Table 7.4 shows that combining our custom augmentation policy (A^+) with the model from Section 7.4.1 results in considerable improvements on all tasks.

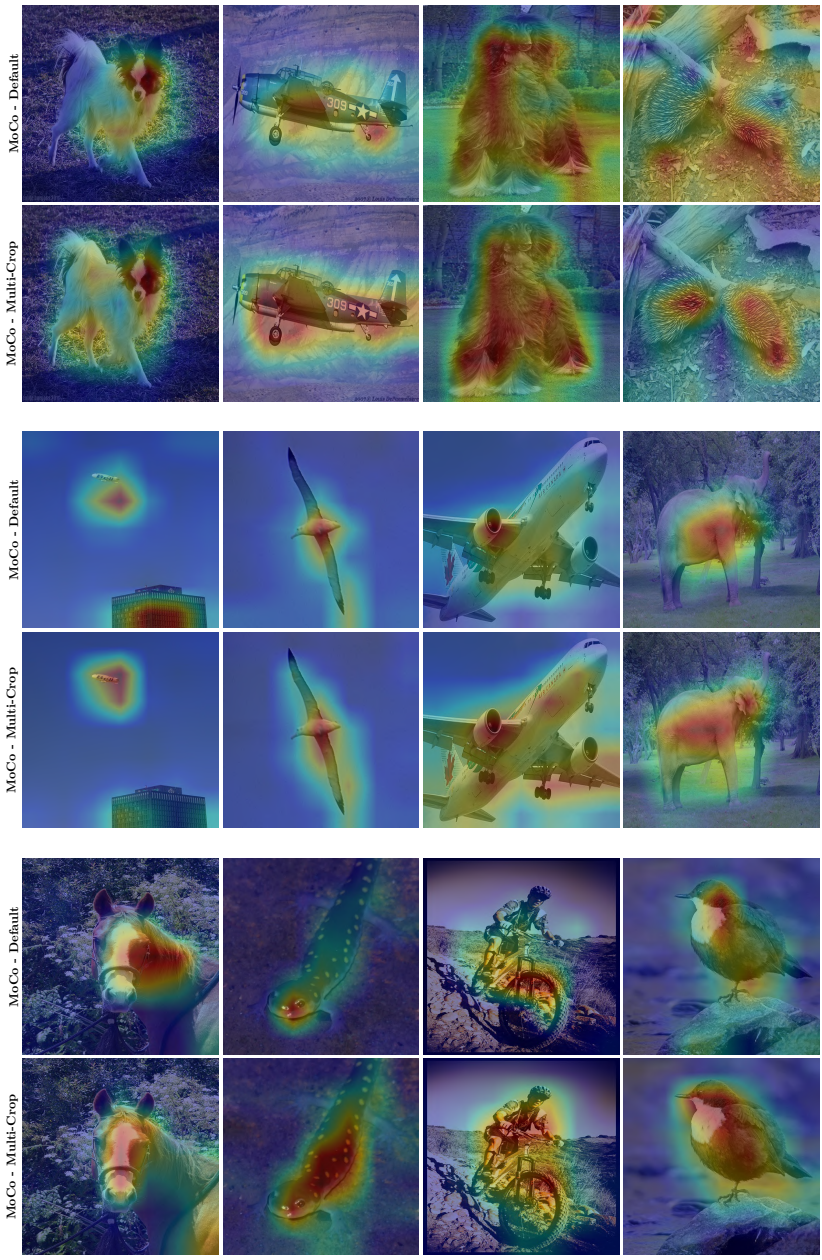


Figure 7.7: Class activation maps for the two-crop versus multi-crop model.

Table 7.5: **DAVIS 2017 video instance segmentation**. We use the publicly available code from [68] to evaluate our frozen representations. Qualitative results are shown for MoCo trained with the `multi-crop` (MC) transform.

Method	**Data**	$\mathcal{J}_m \uparrow$	$\mathcal{F}_m \uparrow$
DenseCL [160]	COCO	60.6	63.9
MAST [83]	YT-VOS	63.3	67.6
STC [68]	Kinetics	64.8	70.2
MoCo	COCO	61.6	66.6
MoCo + MC	COCO	64.3	69.4

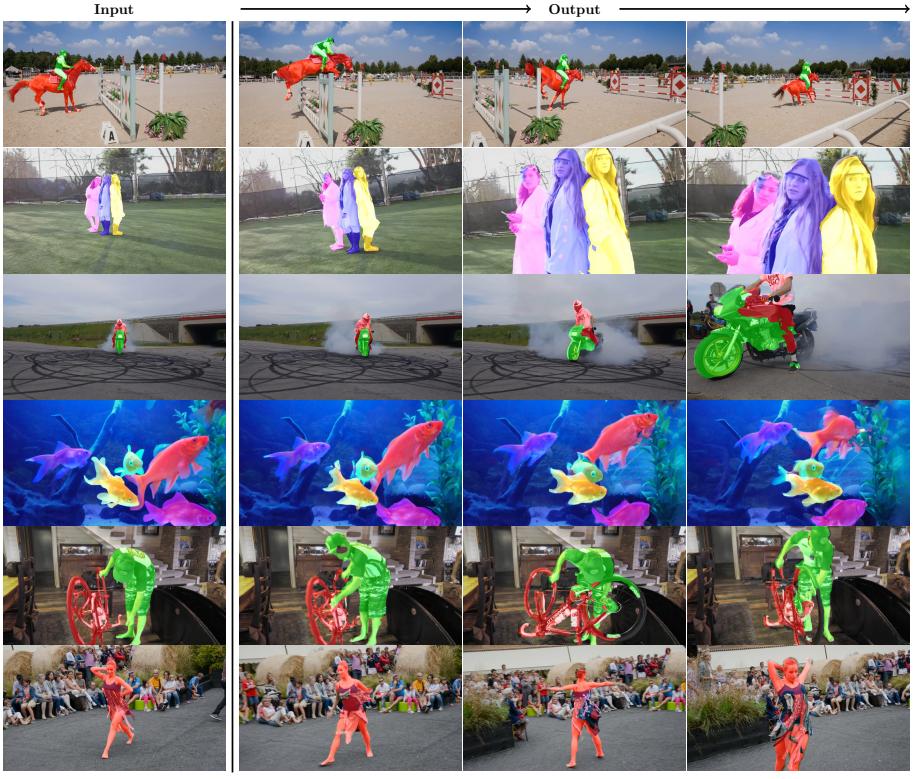


Table 7.6: **VOC semantic segment retrieval.** Each image is partitioned into a number of segments by running K-Means (K=15) on the spatial features. Then we adopt a region descriptor - computed by averaging the embeddings of all pixels within a segment - to obtain nearest neighbors of the validation regions from the train set. Qualitative results are shown for MoCo trained with the `multi-crop` (MC) augmentation strategy (K=5).

	Segm. Retr. VOC	
Method	7 classes (IoU)↑	21 classes (IoU)↑
SegSort [177]	10.2	-
SSL HG [177]	24.6	-
DenseCL [160]	48.4	35.1
MoCo	41.8	28.1
MoCo + MC	48.1	35.1

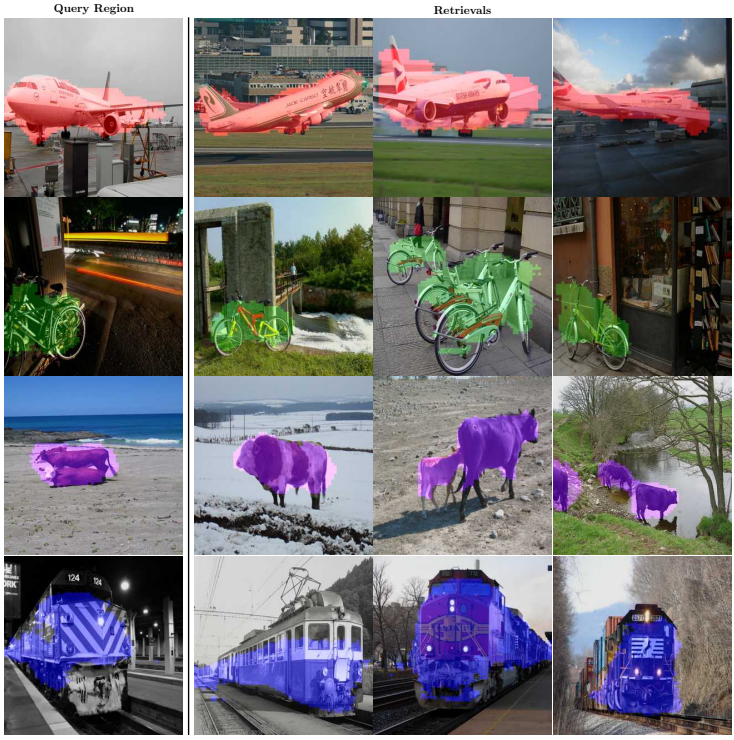


Table 7.7: Ablation of augmentation policies applied to random crops.

Augmentation policy	**VOC SVM** (mAP)
Standard	82.8
AutoAugment	79.9
Standard or AutoAugment	83.7

7.4.3 Nearest Neighbors

Prior work [148] showed that the model learns to map visually similar images closer together than dissimilar ones when tackling the instance discrimination task. In this section, we build upon this insight by imposing additional invariances between neighboring samples. By leveraging other samples as positives, we can capture a rich set of deformations that are hard to model via handcrafted augmentations. However, a well-known problem with methods that group different samples together is preventing representation collapse. Therefore, we formulate our intuition as an auxiliary loss that regularizes the representations - keeping the instance discrimination task as the main force.

Setup Recall that the encoder f consists of a backbone g and a projection head h. MoCo maintains a queue $\mathcal{Q}_h$ of encoded anchors $\{q_0^h \ ... \ q_{K-1}^h\}$ processed by the momentum encoder f'. We now introduce a second - equally sized and aligned - queue $\mathcal{Q}_g$ which maintains the features from before the projection head $\{q_0^g \ ... \ q_{K-1}^g\}$. The queue of backbone features $\mathcal{Q}_g$ can be used to mine nearest neighbors on-the-fly during training. In particular, for a positive x^+ - processed by the encoder f - the set of its k nearest neighbors $\mathcal{N}_{x^+} = \{q_i^h \mid sim(g(x), \ q_i^g) \text{ is top } k \in \mathcal{Q}_g\}$ is computed w.r.t. the queue $\mathcal{Q}_g$. The cosine similarity measure is denoted by sim. Finally, we use the contrastive loss to maximize the agreement between x^+ and its nearest neighbors $\mathcal{N}_{x^+}$ after the projection head:

$$\mathcal{L}_{\text{nn}} = -\sum_{x^+ \in \mathcal{X}^+} \frac{1}{k} \sum_{q \in \mathcal{N}_{x^+}} \frac{\exp\left[q^T \cdot f(x^+)/\tau\right]}{\exp\left[q^T \cdot f(x^+)/\tau\right] + \sum_{x^- \in \mathcal{X}^-} \exp\left[q^T \cdot f(x^-)/\tau\right]}. \tag{7.2}$$

The total loss is the sum of the instance discrimination loss $\mathcal{L}_{\text{inst}}$ and the nearest neighbors loss $\mathcal{L}_{\text{nn}}$ from Eq. 7.2: $\mathcal{L}_{\text{inst}} + \lambda \mathcal{L}_{\text{nn}}$. Figure 7.8 shows a schematic overview of our k-Nearest Neighbors based Momentum Contrast setup (kNN-MoCo). Algorithm 3 contains the pseudocode.

Algorithm 3 Pseudocode of kNN-MoCo in a PyTorch-like style.

```
# g, g_m: the backbone g and momentum updated backbone g_m
# h, h_m: the projection head h and momentum updated projection head h_m (MLP)
# anchors: batch of anchor images (B x 3 x 224 x 224)
# positives: batch of global positive views (B x 3 x 224 x 224)
# positives_small: batch of N local positive views (B*N x 3 x 96 x 96)
# queue: dictionary as a queue (C x K)
# queue_g: dictionary as a queue (C_g x K)
# m: momentum
# t: temperature
# k: number of nearest neighbors
# lambda: weight in range [0, 1]

g_m.params = g.params # initialize momentum updated backbone
h_m.params = h.params # initialize momentum updated head

for batch in loader: # load a minibatch with B samples

   # randomly augment batch
   anchors   = aug(batch)
   positives = aug(batch)
   positives_small = aug_small(batch, anchors) # constrained multi-crop
                Sec. 4.1

   # forward pass
   anchors_g, positives_g = g_m(anchors), g(positives) # B x C_g
   anchors, positives = h_m(anchors_g), h(positives_g) # B x C
   positives_small_g = g(positives_small) # B*N x C_g
   positives_small = h(positives_small_g) # B*N x C

   # concatenate positive views
   positives_g = cat([positives_g, positives_small_g], dim=0) # B*(N+1) x C_g
   positives = cat([positives, positives_small], dim=0) # B*(N+1) x C

   # compute logits
   l_pos = bmm(positives.view(B,N+1,C), anchors.view(B,C,1)) # B x N+1 x 1
   l_neg = mm(positives, queue.view(C,K)).view(B,N+1,K) # B x N+1 x K
   logits = cat([l_pos, l_neg], dim=2) # B x N+1 x K+1

   # determine indices of the nearest neighbors
   indices = topk(mm(positives_g, queue_g).view(B,N+1,K),dim=2) # B x N+1 x k
           Sec. 4.3

   # loss: (1) sharpen with temperature t, (2) apply cross-entropy loss
   loss_inst = CE(logits/t, zeros((B,N+1))
   loss_nn = multi_label_CE(l_neg/t, indices)
   loss = loss_inst + lambda * loss_nn

   # SGD update: g, h
   loss.backward()
   update(g.params, h.params)

   # momentum update: g_m, h_m
   g_m.params = m * g_m.params + (1-m) * g.params
   h_m.params = m * h_m.params + (1-m) * h.params

   # update dictionaries: queue & queue_g
   enqueue_dequeue(queue, anchors)
   enqueue_dequeue(queue_g, anchors_g)
```

`bmm`: batch matrix mult.; `mm`: matrix mult.; `cat`: concatenate; `topk`: indices of top-k largest elements; `CE`: cross-entropy loss
Important differences with MoCo are highlighted.

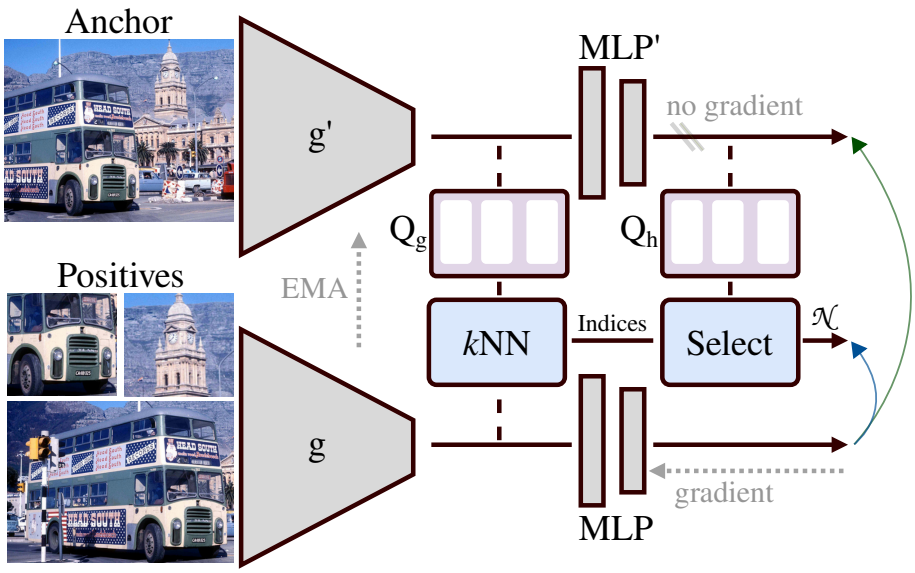


Figure 7.8: **kNN-MoCo setup.** Q_g and Q_h maintain aligned queues of backbone and output features. As before, the encoder f needs to match positives with their anchor (green arrow). Additionally, we also match the positives with their k nearest neighbors obtained from the queue Q_g (blue arrow).

Table 7.8: Ablation study of the number of neighbors k and weight λ for a linear classifier on VOC. $\lambda = 0$ represents the multi-crop model from Section 7.4.1. Models are trained for 200 epochs on COCO.

k	1	5	10	20	50	λ	0.0	0.1	0.2	0.4	0.8
mAP (%)	83.6	84.0	84.1	84.2	84.3	mAP (%)	82.8	83.8	84.1	84.2	80.7

Table 7.9: **State-of-the-art comparison.** MoCo and DenseCL are trained for 800 epochs on COCO. VirTex is trained on COCO captions [24]. MoCo is trained while imposing various additional invariances.

	Semantic seg. (mIoU)			**Classification** (mAP / Acc. / Acc.)			**Vid. Seg.** ($\mathcal{J}\&\mathcal{F}$)	**Depth** (rmse)
Method	**VOC**	**Cityscapes**	**NYUD**	**VOC**	**ImageNet**	**Places**	**DAVIS**	**NYUD**
Rand. init.	39.2	65.0	24.4	-	-	-	40.8	0.867
DenseCL [160]	73.2	**73.5**	**42.1**	83.5	49.9	45.8	61.8	0.589
VirTex [36]	72.7	72.5	40.3	**87.4**	53.8	40.8	61.3	0.613
MoCo	71.1	71.3	40.0	81.0	49.8	44.7	63.3	0.606
+ CC	72.2	71.6	40.4	84.0	54.6	46.1	65.5	0.595
+ CC + A^+	72.7	71.8	40.7	85.0	56.0	47.0	65.7	0.590
+ CC + A^+ + kNN	**73.5**	72.3	41.3	85.9	**56.1**	**48.6**	**66.2**	**0.580**

CC: Constrained multi-crop (Sec. 7.4.1), A^+: Stronger augmentations (Sec. 7.4.2), kNN: nearest neighbors (Sec. 7.4.3).

Results & Discussion. Table 7.8 contains an ablation study of the number of neighbors k and the weight λ. The performance remains stable for a large range of neighbors k (λ is fixed at 0.4). However, increasing the number of neighbors k positively impacts the accuracy. We use $k = 20$ for the remainder of our experiments. Further, the representation quality degrades when using a large weight (e.g. $\lambda = 0.8$). This shows the importance of using the instance discrimination task as our main objective. Also, not shown in the table, we found that it is important to mine the neighbors using the features before the projection head (84.2% vs 82.8%). Finally, Table 7.4 shows improved results on all tasks when combining the nearest neighbors loss with our other modifications. In conclusion, we have successfully explored the data manifold to learn additional invariances. The proposed implementation can be seen as a simple alternative to clustering-based methods [7, 16, 88, 159].

7.4.4 Discussion

We retrain our final model for 800 epochs on COCO and compare with two other methods, i.e. DenseCL [160] and Virtex [36]. We draw the following conclusions from the results in Table 7.9. First, our model improves over the MoCo baseline. The proposed modifications force the model to learn useful features that can not be learned through standard data augmentations, even when increasing the training time. Second, our representations outperform other works on several downstream tasks. These frameworks used more advanced schemes [160] or caption annotations [36]. Interestingly, DenseCL reports better results for the segmentation tasks on Cityscapes and NYUD, but performs worse on other tasks. In contrast, the performance of our representations is better balanced across tasks. We conclude that generic pretraining is still an unsolved problem.

7.5 Related Work

Contrastive learning The idea of contrastive learning [54, 120] is to attract positive sample pairs and repel negative sample pairs. Self-supervised methods [21, 50, 56, 115, 146, 147, 162, 170, 25] have used the contrastive loss to learn visual representations from unlabeled images. Augmentations of the same image are used as positives, while other images are considered as negatives.

A number of extensions were proposed to boost the performance. For example, a group of works [160, 126, 150, 60] applied the contrastive loss at the pixel-level to learn dense representations. Others improved the representations for object recognition tasks by re-identifying patches [168] or by maximizing the similarity of corresponding image regions in the intermediate network layers [164]. Finally, Selvaraju *et al.* [138]

employed an attention mechanism to improve the visual grounding abilities of the model. In contrast to these works, we do not employ a more advanced pretext task to learn spatially structured representations. Instead, we adopt a standard framework [56] and find that the learned representations exhibit similar properties when modifying the cropping strategy. Further, we expect that our findings can be relevant for other contrastive learning frameworks too.

Clustering Several works combined clustering with self-labeling [15, 7] or contrastive learning [16, 88, 159] to learn representations in a self-supervised way. Similar to the nearest neighbors loss (Eq. 7.2), these frameworks explore the data manifold to learn invariances. Differently, we avoid the use of a clustering criterion like K-Means by computing nearest neighbors on-the-fly w.r.t. a memory bank. A few other works [65, 148] also used nearest neighbors as positive pairs in an auxiliary loss. However, the neighbors had to be computed off-line at fixed intervals during training. Concurrent to our work, Dwibedi *et al.* [39] adopted nearest neighbors from a memory bank under the BYOL [50] framework. The authors focus on image classification datasets. In conclusion, we propose a simple, yet effective alternative to existing clustering methods.

Other Contrastive SSL has been the subject of several recent surveys [128, 181, 44]. We list the most relevant ones. Similar to our work, Zhao *et al.* [181] pretrain on multiple datasets. They investigate what information is retained under the transfer learning setup, which differs from the focus of this paper. Purushwalkam and Gupta [128] study the influence of the object-centric dataset bias, but their experimental scope is rather limited, and their conclusions diverge from the ones in this work. Ericsson *et al.* [44] compare several ImageNet pretrained models under the transfer learning setup. In conclusion, we believe our study can complement these works.

7.6 Application: Indoor Scene Understanding

In this section, we apply self-supervised pretraining to improve a model for indoor scene understanding. We consider the semantic segmentation and depth estimation tasks on the NYUD dataset [140]. The model consists of a ResNet-50 backbone with ASPP head. Table 7.10 shows the results obtained via single-task learning and a multi-task learning baseline. In both cases, we obtain better results when using MoCo pretrained weights versus supervised pretraining. We conclude that the gains obtained with contrastive self-supervised learning translate to multi-task learning models as well.

Table 7.10: Comparison of pretraining strategies under a single- and multi-task learning setup on NYUD. We consider the tasks of semantic segmentation and monocular depth estimation. The multi-task learning model with MoCo pretrained weights reports the best results on both tasks.

Setup	**Pretraining (ImageNet)**	**Sem. Seg.** (IoU)	**Depth** (rmse)
Single-Task Learning	Supervised	43.6	0.599
	MoCo	44.7	0.590
Multi-Task Learning	Supervised	43.7	0.593
	MoCo	**45.2**	**0.581**

7.7 Conclusion

In this chapter, we showed that we can find a generic set of augmentations/invariances that allows us to learn effective representations across different types of datasets (i.e., scene-centric, non-uniform, domain-specific). We provide empirical evidence to support this claim. First, in Section 7.3, we show that the standard SimCLR augmentations can be applied across several datasets. Then, Section 7.4 studies the use of additional invariances to improve the results for a generic dataset (i.e., MS-COCO). In this way, we reduce the need for dataset or domain-specific expertise to learn useful representations in a self-supervised way. Instead, the results show that simple contrastive frameworks apply to a wide range of datasets. We believe this is an encouraging result. Finally, our overall conclusion differs from a few recent works [128, 138], which investigated the use of a more advanced pretext task or video to learn visual representations.

Our results also yields a few interesting follow-up questions. **Modalities.** Can we reach similar conclusions for other modalities like video, text, audio, etc.? **Invariances.** What other invariances can be applied? How do we bias the representations to focus more on texture, shape or other specific properties? **Compositionality.** Can we combine different datasets to learn better representations?

Limitations: We performed extensive experiments to measure the influence of different dataset biases on the representations. The experimental setup covered the use of object-centric vs. scene-centric data, uniform vs. long-tailed class distributions and general vs. domain-specific data. Further, we explored various ways of imposing additional invariances to improve the representations. Undoubtedly, there are several components that fall outside the scope of our study. We briefly discuss some of the limitations below.

Dataset size. The experiments were conducted on datasets of moderate size (e.g. MS-COCO contains 118K images). It remains an open question whether the same results will be observed for larger datasets. We believe it would be useful to further investigate the behavior of existing methods on large-scale datasets.

Data augmentations. Imposing invariances to different data transformations proves crucial to learn useful representations. In this work, we used the same data augmentation strategy for all our experiments. However, different types of data could benefit from a different set of augmentations. We hypothesize that this is particularly true for domain-specific datasets. In this case, we expect that specialized data transformations, based upon domain-knowledge, could further boost the results. Further, we observe that the performance of existing methods still strongly depends on handcrafted augmentations, which can limit their applicability. We tried to partially alleviate this problem by adopting nearest neighbors - which only relies on the data manifold itself. Still, this problem could benefit from further investigation.

Inductive biases. All experiments were performed with a ResNet architecture. It would be interesting to study how architectural design choices influence the representation quality. In particular, one could study more general models like transformers [38, 155] which incorporate different biases.

Chapter 8

Conclusion

In this manuscript, we have proposed several techniques for improving deep multi-task learning models in computer vision. This chapter gives an overview of the contributions of each chapter. Further, we also address the limitations of our work and discuss possible directions for future research.

8.1 Summary of Contributions

In Chapter 3, we reviewed recent methods for MTL within the scope of deep neural networks. We presented an extensive overview of both architectural and optimization based strategies for MTL. For each method, we described its key aspects, discussed the commonalities and differences with related works, and presented the possible advantages or disadvantages. Further, we introduced a novel taxonomy for describing deep multi-task learning architectures. The proposed taxonomy discriminates between encoder- and decoder-focused models based on where the task interactions are modeled. Encoder-focused models only share information during the encoding stage, while the decoding process happens in an isolated task-specific manner. Differently, decoder-focused models first make initial task predictions, and then leverage features from these initial predictions to further improve each task output - in a one-off or recursive manner. Finally, starting from our literature review, we motivated the various lenses through which we studied the multi-task learning problem in this manuscript.

In Chapter 4, we considered the use of branched multi-task networks in which deeper layers gradually grow more task-specific. Additionally, we introduced a principled method to automate the construction of such branched multi-task networks. The

construction process groups related tasks together while making a trade-off between task similarity and network complexity. The task relatedness scores are based on the premise that similar tasks can be solved with a similar set of features. The proposed approach was evaluated on three different benchmarks including CelebA, Cityscapes and Taskonomy. The results show that, for a given parameter or FLOPS budget, our method consistently yields models with the highest performance.

Chapter 5 continued on improving the architecture. This time, we focused on building better decoder-focused models for tackling dense prediction tasks. First, we argued about the importance of modeling task interactions at multiple scales when distilling the task information under a multi-task learning setup. Guided by this principle, we then proposed a novel architecture named MTI-Net. Experiments on two multi-task dense labeling datasets, i.e. NYUD and PASCAL, showed that our model delivers on the full potential of multi-task learning, that is, smaller memory footprint, reduced number of calculations, and better performance w.r.t single-task learning.

Chapter 6 studied the problem of balancing the learning of tasks under a MTL setup. In particular, we need to avoid that one or more tasks dominate the weight update, thereby hindering the model from learning the other tasks. We analyzed multiple existing task balancing strategies and talked about their commonalities and differences. Additionally, we carried out an ablation study to isolate the elements that are most effective. Based upon our results, we proposed a set of heuristics to help practitioners determine the task-specific weights. Applying the heuristics results in more robust performance across different datasets. This differs from existing methods, which seem tuned for specific scenarios.

Chapter 7 considered the use of self-supervised pretraining to improve models for scene understanding. First, we showed that an existing technique like MoCo can be used for pretraining across a variety of datasets including scene-centric, long-tailed and domain-specific data. Second, we showed that the representations can be improved by imposing additional invariances. This result directly benefits many downstream tasks like semantic segmentation, detection, etc. Finally, we showed that the improvements obtained with self-supervised pretraining also translate to multi-task learning models.

We can also draw more general conclusions from this work. First, it requires some effort to get the full benefits from multi-task learning: reduced memory footprint, fewer computations and improved performance. In particular, the improvements obtained with a simple multi-task learning baseline seem limited to a few specific, isolated cases like multi-attribute learning, or specific combinations of tasks like semantic segmentation and depth. It pays off to use more specialized models like MTI-Net when it comes to tackling multiple dense prediction tasks. Second, balancing the learning of tasks still requires a significant amount of trial-and-error. Many existing methods were designed for specific cases, e.g. uncertainty weighting works well with noisy data. We have addressed this problem by proposing a set of heuristics, which can be

useful for practitioners working on applications. Third, contrastive self-supervised learning emerges as a powerful technique for improving scene understanding models. We believe such pretraining techniques will quickly get adopted, as they have the potential to benefit applications were annotated data is scarce, like medical imaging, or where large amounts of unlabeled data are readily available, like autonomous driving.

8.2 Limitations and Future Work

Undoubtedly, there are several aspects of the problem that fall outside the scope of this manuscript. The limitations tied to the proposed methods were already discussed at the end of every chapter. Therefore, we focus on some of the more general limitations below.

First, the software systems that implement state-of-the-art algorithms for tackling tasks like object detection, semantic segmentation, depth estimation, etc. are well established. Examples include Detectron [161], TorchVision [123], MMLab [18], etc. The availability of these frameworks allows rapid progress and a thorough evaluation of novel research contributions. Unfortunately, at this time, there are no such software frameworks for training multi-task learning models. As a result, different groups of researchers have implemented their own dataloaders, evaluation criteria, baselines, etc., which makes it hard to draw apples-to-apples comparisons between works. We partially addressed this problem in our survey paper [153]. Still, we believe the community could benefit from the development of a software framework with more explicit support for multi-task learning. This would not only allow to benchmark progress in the field more easily, but also be useful for engineers that are building applications.

Second, very few works have considered MTL under the semi-supervised or active learning setting. Nonetheless, we believe that these are interesting directions for future research. For example, a major limitation of the fully-supervised MTL setting that we considered is the requirement for all samples to be annotated for every task. Although a few works [74, 116] have already touched upon this problem, we believe that this is an interesting direction for future work.

Third, in this work we have focused on tackling multiple visual recognition tasks using images. It would be interesting to also include other modalities like text, video, audio, etc. A few works [106, 64] have already shown promising results for such multi-modal learning problems. Further research along this direction could help to build more general purpose intelligence agents capable of handling larger task dictionaries across multiple domains.

Finally, multi-task learning was recently shown to improve robustness. For example, in [111] a multi-task learning strategy showed robustness against adversarial attacks,

while [175] found that applying cross-task consistency in MTL improves generalization, and allows for domain shift detection. Similarly, the performance profiles in Chapter 6 showed that MTL algorithms are more robust against changes in the hyperparameter settings than others. It could be useful to further investigate this behavior.

Bibliography

[1] ACHILLE, A., LAM, M., TEWARI, R., RAVICHANDRAN, A., MAJI, S., FOWLKES, C. C., SOATTO, S., AND PERONA, P. Task2vec: Task embedding for meta-learning. In *ICCV* (2019).

[2] AGARWAL, A., GERBER, S., AND DAUME, H. Learning multiple tasks using manifold regularization. In *NeurIPS* (2010).

[3] AMODEI, D., ANANTHANARAYANAN, S., ANUBHAI, R., BAI, J., BATTENBERG, E., CASE, C., CASPER, J., CATANZARO, B., CHENG, Q., CHEN, G., ET AL. Deep speech 2: End-to-end speech recognition in english and mandarin. In *ICML* (2016).

[4] ANDO, R. K., AND ZHANG, T. A framework for learning predictive structures from multiple tasks and unlabeled data. *JMLR* (2005).

[5] ARGYRIOU, A., EVGENIOU, T., AND PONTIL, M. Convex multi-task feature learning. *Machine learning* (2008).

[6] ASANO, Y. M., PATRICK, M., RUPPRECHT, C., AND VEDALDI, A. Labelling unlabelled videos from scratch with multi-modal self-supervision. In *NeurIPS* (2020).

[7] ASANO, Y. M., RUPPRECHT, C., AND VEDALDI, A. Self-labelling via simultaneous clustering and representation learning. In *ICLR* (2020).

[8] BAKKER, B., AND HESKES, T. Task clustering and gating for bayesian multitask learning. *JMLR* (2003).

[9] BANSAL, A., CHEN, X., RUSSELL, B., GUPTA, A., AND RAMANAN, D. Pixelnet: Representation of the pixels, by the pixels, and for the pixels. *arXiv preprint arXiv:1702.06506* (2017).

[10] BELL, S., LIU, Y., ALSHEIKH, S., TANG, Y., PIZZI, E., HENNING, M., SINGH, K., PARKHI, O., AND BORISYUK, F. Groknet: Unified computer vision model trunk and embeddings for commerce. In *SIGKDD* (2020).

[11] BILEN, H., AND VEDALDI, A. Integrator nets. In *NeurIPS* (2016).

[12] BORSE, S., WANG, Y., ZHANG, Y., AND PORIKLI, F. Inverseform: A loss function for structured boundary-aware segmentation.

[13] BRAGMAN, F. J., TANNO, R., OURSELIN, S., ALEXANDER, D. C., AND CARDOSO, M. J. Stochastic filter groups for multi-task cnns: Learning specialist and generalist convolution kernels. In *ICCV* (2019).

[14] BRUGGEMANN, D., KANAKIS, M., GEORGOULIS, S., AND GOOL, L. V. Automated search for resource-efficient branched multi-task networks. In *BMVC* (2020).

[15] CARON, M., BOJANOWSKI, P., JOULIN, A., AND DOUZE, M. Deep clustering for unsupervised learning of visual features. In *ECCV* (2018).

[16] CARON, M., MISRA, I., MAIRAL, J., GOYAL, P., BOJANOWSKI, P., AND JOULIN, A. Unsupervised learning of visual features by contrasting cluster assignments. In *NeurIPS* (2020).

[17] CARUANA, R. Multitask learning. *Machine learning* (1997).

[18] CHEN, K., WANG, J., PANG, J., CAO, Y., XIONG, Y., LI, X., SUN, S., FENG, W., LIU, Z., XU, J., ZHANG, Z., CHENG, D., ZHU, C., CHENG, T., ZHAO, Q., LI, B., LU, X., ZHU, R., WU, Y., DAI, J., WANG, J., SHI, J., OUYANG, W., LOY, C. C., AND LIN, D. MMDetection: Open mmlab detection toolbox and benchmark. *arXiv preprint arXiv:1906.07155* (2019).

[19] CHEN, L.-C., PAPANDREOU, G., KOKKINOS, I., MURPHY, K., AND YUILLE, A. L. Deeplab: Semantic image segmentation with deep convolutional nets, atrous convolution, and fully connected crfs. *TPAMI* (2017).

[20] CHEN, L.-C., ZHU, Y., PAPANDREOU, G., SCHROFF, F., AND ADAM, H. Encoder-decoder with atrous separable convolution for semantic image segmentation. In *ECCV* (2018).

[21] CHEN, T., KORNBLITH, S., NOROUZI, M., AND HINTON, G. A simple framework for contrastive learning of visual representations. In *ICML* (2020).

[22] CHEN, T., KORNBLITH, S., SWERSKY, K., NOROUZI, M., AND HINTON, G. Big self-supervised models are strong semi-supervised learners. In *NeurIPS* (2020).

[23] CHEN, X., FAN, H., GIRSHICK, R., AND HE, K. Improved baselines with momentum contrastive learning. *arXiv preprint arXiv:2003.04297* (2020).

[24] CHEN, X., FANG, H., LIN, T.-Y., VEDANTAM, R., GUPTA, S., DOLLÁR, P., AND ZITNICK, C. L. Microsoft coco captions: Data collection and evaluation server. *arXiv preprint arXiv:1504.00325* (2015).

[25] CHEN, X., AND HE, K. Exploring simple siamese representation learning. In *CVPR* (2021).

[26] CHEN, X., MOTTAGHI, R., LIU, X., FIDLER, S., URTASUN, R., AND YUILLE, A. Detect what you can: Detecting and representing objects using holistic models and body parts. In *CVPR* (2014).

[27] CHEN, Z., BADRINARAYANAN, V., LEE, C.-Y., AND RABINOVICH, A. Gradnorm: Gradient normalization for adaptive loss balancing in deep multitask networks. In *ICML* (2018).

[28] CHEN, Z., NGIAM, J., HUANG, Y., LUONG, T., KRETZSCHMAR, H., CHAI, Y., AND ANGUELOV, D. Just pick a sign: Optimizing deep multitask models with gradient sign dropout. *NeurIPS* (2020).

[29] CORDTS, M., OMRAN, M., RAMOS, S., REHFELD, T., ENZWEILER, M., BENENSON, R., FRANKE, U., ROTH, S., AND SCHIELE, B. The cityscapes dataset for semantic urban scene understanding. In *CVPR* (2016).

[30] CUBUK, E. D., ZOPH, B., MANE, D., VASUDEVAN, V., AND LE, Q. V. Autoaugment: Learning augmentation strategies from data. In *CVPR* (2019).

[31] CUI, Y., SONG, Y., SUN, C., HOWARD, A., AND BELONGIE, S. Large scale fine-grained categorization and domain-specific transfer learning. In *CVPR* (2018).

[32] DAUMÉ III, H. Bayesian multitask learning with latent hierarchies. *arXiv preprint arXiv:0907.0783* (2009).

[33] DENG, J., DONG, W., SOCHER, R., LI, L.-J., LI, K., AND FEI-FEI, L. Imagenet: A large-scale hierarchical image database. In *CVPR* (2009).

[34] DERUYTTERE, T., VANDENHENDE, S., GRUJICIC, D., LIU, Y., VAN GOOL, L., BLASCHKO, M., TUYTELAARS, T., AND MOENS, M.-F. Commands 4 autonomous vehicles (c4av) workshop summary. In *ECCV* (2020).

[35] DERUYTTERE, T., VANDENHENDE, S., GRUJICIC, D., VAN GOOL, L., AND MOENS, M. F. Talk2car: Taking control of your self-driving car. In *EMNLP-IJCNLP* (2019).

[36] DESAI, K., AND JOHNSON, J. Virtex: Learning visual representations from textual annotations. In *CVPR* (2021).

[37] DÉSIDÉRI, J.-A. Multiple-gradient descent algorithm (mgda) for multiobjective optimization. *Comptes Rendus Mathematique* (2012).

[38] DOSOVITSKIY, A., BEYER, L., KOLESNIKOV, A., WEISSENBORN, D., ZHAI, X., UNTERTHINER, T., DEHGHANI, M., MINDERER, M., HEIGOLD, G., GELLY, S., ET AL. An image is worth 16x16 words: Transformers for image recognition at scale. In *ICLR* (2021).

[39] DWIBEDI, D., AYTAR, Y., TOMPSON, J., SERMANET, P., AND ZISSERMAN, A. With a little help from my friends: Nearest-neighbor contrastive learning of visual representations. *arXiv preprint arXiv:2104.14548* (2021).

[40] DWIVEDI, K., AND ROIG, G. Representation similarity analysis for efficient task taxonomy & transfer learning. In *CVPR* (2019).

[41] EIGEN, D., AND FERGUS, R. Predicting depth, surface normals and semantic labels with a common multi-scale convolutional architecture. In *ICCV* (2015).

[42] EIGEN, D., PUHRSCH, C., AND FERGUS, R. Depth map prediction from a single image using a multi-scale deep network. In *NeurIPS* (2014).

[43] ELSKEN, T., METZEN, J. H., AND HUTTER, F. Neural architecture search: A survey. *JMLR* (2019).

[44] ERICSSON, L., GOUK, H., AND HOSPEDALES, T. M. How well do self-supervised models transfer? In *CVPR* (2021).

[45] EVERINGHAM, M., VAN GOOL, L., WILLIAMS, C. K., WINN, J., AND ZISSERMAN, A. The pascal visual object classes (voc) challenge. *IJCV* (2010).

[46] EVGENIOU, T., AND PONTIL, M. Regularized multi–task learning. In *KDD* (2004).

[47] GAO, Y., MA, J., ZHAO, M., LIU, W., AND YUILLE, A. L. Nddr-cnn: Layerwise feature fusing in multi-task cnns by neural discriminative dimensionality reduction. In *CVPR* (2019).

[48] GONG, T., LEE, T., STEPHENSON, C., RENDUCHINTALA, V., PADHY, S., NDIRANGO, A., KESKIN, G., AND ELIBOL, O. H. A comparison of loss weighting strategies for multi task learning in deep neural networks. *IEEE Access* (2019).

[49] GOODFELLOW, I., BENGIO, Y., COURVILLE, A., AND BENGIO, Y. *Deep learning*. MIT press Cambridge, 2016.

[50] GRILL, J.-B., STRUB, F., ALTCHÉ, F., TALLEC, C., RICHEMOND, P. H., BUCHATSKAYA, E., DOERSCH, C., PIRES, B. A., GUO, Z. D., AZAR, M. G., ET AL. Bootstrap your own latent: A new approach to self-supervised learning. In *NeurIPS* (2020).

[51] GUO, M., HAQUE, A., HUANG, D.-A., YEUNG, S., AND FEI-FEI, L. Dynamic task prioritization for multitask learning. In *ECCV* (2018).

[52] GUO, P., LEE, C.-Y., AND ULBRICHT, D. Learning to branch for multi-task learning. In *ICML* (2020).

[53] GUR, M., AND SNODDERLY, D. M. Direction selectivity in v1 of alert monkeys: evidence for parallel pathways for motion processing. *The Journal of physiology* (2007).

[54] GUTMANN, M., AND HYVÄRINEN, A. Noise-contrastive estimation: A new estimation principle for unnormalized statistical models. In *AISTATS* (2010).

[55] HAND, E. M., AND CHELLAPPA, R. Attributes for improved attributes: A multi-task network utilizing implicit and explicit relationships for facial attribute classification. In *AAAI* (2017).

[56] HE, K., FAN, H., WU, Y., XIE, S., AND GIRSHICK, R. Momentum contrast for unsupervised visual representation learning. In *CVPR* (2020).

[57] HE, K., GKIOXARI, G., DOLLÁR, P., AND GIRSHICK, R. Mask r-cnn. In *ICCV* (2017).

[58] HE, K., ZHANG, X., REN, S., AND SUN, J. Spatial pyramid pooling in deep convolutional networks for visual recognition. *TPAMI* (2015).

[59] HE, K., ZHANG, X., REN, S., AND SUN, J. Deep residual learning for image recognition. In *CVPR* (2016).

[60] HÉNAFF, O. J., KOPPULA, S., ALAYRAC, J.-B., OORD, A. V. D., VINYALS, O., AND CARREIRA, J. Efficient visual pretraining with contrastive detection. *arXiv preprint arXiv:2103.10957* (2021).

[61] HIRSCHMULLER, H. Stereo processing by semiglobal matching and mutual information. *TPAMI* (2008).

[62] HOIEM, D., EFROS, A. A., AND HEBERT, M. Closing the loop in scene interpretation. In *CVPR* (2008), IEEE.

[63] HU, J., SHEN, L., AND SUN, G. Squeeze-and-excitation networks. In *CVPR* (2018).

[64] HU, R., AND SINGH, A. Unit: Multimodal multitask learning with a unified transformer. *arXiv preprint arXiv:2102.10772* (2021).

[65] HUANG, J., DONG, Q., GONG, S., AND ZHU, X. Unsupervised deep learning by neighbourhood discovery. In *ICML* (2019).

[66] HUANG, S., LI, X., CHENG, Z., HAUPTMANN, A., ET AL. Gnas: A greedy neural architecture search method for multi-attribute learning. In *ACMMM* (2018).

[67] HWANG, J.-J., YU, S. X., SHI, J., COLLINS, M. D., YANG, T.-J., ZHANG, X., AND CHEN, L.-C. Segsort: Segmentation by discriminative sorting of segments. In *CVPR* (2019).

[68] JABRI, A., OWENS, A., AND EFROS, A. A. Space-time correspondence as a contrastive random walk. In *NeurIPS* (2020).

[69] JACOB, L., VERT, J.-P., AND BACH, F. R. Clustered multi-task learning: A convex formulation. In *NeurIPS* (2009).

[70] JALALI, A., SANGHAVI, S., RUAN, C., AND RAVIKUMAR, P. K. A dirty model for multi-task learning. In *NeurIPS* (2010).

[71] JING, L., AND TIAN, Y. Self-supervised visual feature learning with deep neural networks: A survey. *TPAMI* (2020).

[72] KENDALL, A., BADRINARAYANAN, V., AND CIPOLLA, R. Bayesian segnet: Model uncertainty in deep convolutional encoder-decoder architectures for scene understanding. In *BMVC* (2017).

[73] KENDALL, A., GAL, Y., AND CIPOLLA, R. Multi-task learning using uncertainty to weigh losses for scene geometry and semantics. In *CVPR* (2018).

[74] KIM, D.-J., CHOI, J., OH, T.-H., YOON, Y., AND KWEON, I. S. Disjoint multi-task learning between heterogeneous human-centric tasks. In *WACV* (2018).

[75] KIRILLOV, A., GIRSHICK, R., HE, K., AND DOLLÁR, P. Panoptic feature pyramid networks. In *CVPR* (2019).

[76] KOKKINOS, I. Pushing the boundaries of boundary detection using deep learning. *arXiv preprint arXiv:1511.07386* (2015).

[77] KOKKINOS, I. Ubernet: Training a universal convolutional neural network for low-, mid-, and high-level vision using diverse datasets and limited memory. In *CVPR* (2017).

[78] KRAUSE, J., DENG, J., STARK, M., AND FEI-FEI, L. Collecting a large-scale dataset of fine-grained cars.

[79] KRIEGESKORTE, N., MUR, M., AND BANDETTINI, P. A. Representational similarity analysis-connecting the branches of systems neuroscience. *Frontiers in systems neuroscience* (2008).

[80] KRIZHEVSKY, A., HINTON, G., ET AL. Learning multiple layers of features from tiny images.

[81] KUMAR, A., AND DAUME III, H. Learning task grouping and overlap in multi-task learning. In *ICML* (2012).

[82] KUZNETSOVA, A., ROM, H., ALLDRIN, N., UIJLINGS, J., KRASIN, I., PONT-TUSET, J., KAMALI, S., POPOV, S., MALLOCI, M., KOLESNIKOV, A., ET AL. The open images dataset v4. *IJCV* (2020).

[83] LAI, Z., LU, E., AND XIE, W. Mast: A memory-augmented self-supervised tracker. In *CVPR* (2020).

[84] LECUN, Y., AND BENGIO, Y. Convolutional networks for images, speech, and time series. In *The handbook of brain theory and neural networks*. 1998.

[85] LECUN, Y., BOSER, B., DENKER, J. S., HENDERSON, D., HOWARD, R. E., HUBBARD, W., AND JACKEL, L. D. Backpropagation applied to handwritten zip code recognition. *Neural computation* (1989).

[86] LEE, S.-I., CHATALBASHEV, V., VICKREY, D., AND KOLLER, D. Learning a meta-level prior for feature relevance from multiple related tasks. In *ICML* (2007).

[87] LI, B., SHEN, C., DAI, Y., VAN DEN HENGEL, A., AND HE, M. Depth and surface normal estimation from monocular images using regression on deep features and hierarchical crfs. In *CVPR* (2015).

[88] LI, J., ZHOU, P., XIONG, C., SOCHER, R., AND HOI, S. C. Prototypical contrastive learning of unsupervised representations. In *ICLR* (2021).

[89] LIANG, J., MEYERSON, E., AND MIIKKULAINEN, R. Evolutionary architecture search for deep multitask networks. In *GECCO* (2018).

[90] LIN, G., MILAN, A., SHEN, C., AND REID, I. Refinenet: Multi-path refinement networks for high-resolution semantic segmentation. In *CVPR* (2017).

[91] LIN, G., SHEN, C., VAN DEN HENGEL, A., AND REID, I. Efficient piecewise training of deep structured models for semantic segmentation. In *CVPR* (2016).

[92] LIN, T.-Y., DOLLÁR, P., GIRSHICK, R., HE, K., HARIHARAN, B., AND BELONGIE, S. Feature pyramid networks for object detection. In *CVPR* (2017).

[93] LIN, T.-Y., GOYAL, P., GIRSHICK, R., HE, K., AND DOLLÁR, P. Focal loss for dense object detection. In *ICCV* (2017).

[94] LIN, T.-Y., MAIRE, M., BELONGIE, S., HAYS, J., PERONA, P., RAMANAN, D., DOLLÁR, P., AND ZITNICK, C. L. Microsoft coco: Common objects in context. In *ECCV* (2014).

[95] LIN, X., ZHEN, H.-L., LI, Z., ZHANG, Q.-F., AND KWONG, S. Pareto multi-task learning. In *NeurIPS* (2019).

[96] LIU, C., ZOPH, B., NEUMANN, M., SHLENS, J., HUA, W., LI, L.-J., FEI-FEI, L., YUILLE, A., HUANG, J., AND MURPHY, K. Progressive neural architecture search. In *ECCV* (2018).

[97] LIU, F., SHEN, C., LIN, G., AND REID, I. Learning depth from single monocular images using deep convolutional neural fields. *TPAMI* (2015).

[98] LIU, H., SIMONYAN, K., VINYALS, O., FERNANDO, C., AND KAVUKCUOGLU, K. Hierarchical representations for efficient architecture search. In *ICLR* (2018).

[99] LIU, J., JI, S., AND YE, J. Multi-task feature learning via efficient l2, 1-norm minimization. In *Uncertainty in Artificial Intelligence* (2009).

[100] LIU, P., QIU, X., AND HUANG, X. Adversarial multi-task learning for text classification. In *ACL* (2017).

[101] LIU, S., JOHNS, E., AND DAVISON, A. J. End-to-end multi-task learning with attention. In *CVPR* (2019).

[102] LIU, Z., LUO, P., WANG, X., AND TANG, X. Deep learning face attributes in the wild. In *ICCV* (2015).

[103] LIU, Z., MIAO, Z., ZHAN, X., WANG, J., GONG, B., AND YU, S. X. Large-scale long-tailed recognition in an open world. In *CVPR* (2019).

[104] LONG, J., SHELHAMER, E., AND DARRELL, T. Fully convolutional networks for semantic segmentation. In *CVPR* (2015).

[105] LONG, M., CAO, Z., WANG, J., AND PHILIP, S. Y. Learning multiple tasks with multilinear relationship networks. In *NeurIPS* (2017).

[106] LU, J., GOSWAMI, V., ROHRBACH, M., PARIKH, D., AND LEE, S. 12-in-1: Multi-task vision and language representation learning. In *CVPR* (2020).

[107] LU, Y., KUMAR, A., ZHAI, S., CHENG, Y., JAVIDI, T., AND FERIS, R. Fully-adaptive feature sharing in multi-task networks with applications in person attribute classification. In *CVPR* (2017).

[108] MALLYA, A., DAVIS, D., AND LAZEBNIK, S. Piggyback: Adapting a single network to multiple tasks by learning to mask weights. In *ECCV* (2018).

[109] MANINIS, K.-K., PONT-TUSET, J., ARBELÁEZ, P., AND VAN GOOL, L. Convolutional oriented boundaries: From image segmentation to high-level tasks. *TPAMI* (2017).

[110] MANINIS, K.-K., RADOSAVOVIC, I., AND KOKKINOS, I. Attentive single-tasking of multiple tasks. In *CVPR* (2019).

[111] MAO, C., GUPTA, A., NITIN, V., RAY, B., SONG, S., YANG, J., AND VONDRICK, C. Multitask learning strengthens adversarial robustness. In *ECCV* (2020).

[112] MARTIN, D. R., FOWLKES, C. C., AND MALIK, J. Learning to detect natural image boundaries using local brightness, color, and texture cues. *TPAMI* (2004).

[113] MEYERSON, E., AND MIIKKULAINEN, R. Beyond shared hierarchies: Deep multitask learning through soft layer ordering. In *ICLR* (2018).

[114] MISRA, I., SHRIVASTAVA, A., GUPTA, A., AND HEBERT, M. Cross-stitch networks for multi-task learning. In *CVPR* (2016).

[115] MISRA, I., AND VAN DER MAATEN, L. Self-supervised learning of pretext-invariant representations. In *CVPR* (2020).

[116] NEKRASOV, V., DHARMASIRI, T., SPEK, A., DRUMMOND, T., SHEN, C., AND REID, I. Real-time joint semantic segmentation and depth estimation using asymmetric annotations. In *ICRA* (2019).

[117] NEVEN, D., DE BRABANDERE, B., GEORGOULIS, S., PROESMANS, M., AND VAN GOOL, L. Fast scene understanding for autonomous driving. In *IV Workshops* (2017).

[118] NEWELL, A., JIANG, L., WANG, C., LI, L.-J., AND DENG, J. Feature partitioning for efficient multi-task architectures. *arXiv preprint arXiv:1908.04339* (2019).

[119] NEWELL, A., YANG, K., AND DENG, J. Stacked hourglass networks for human pose estimation. In *ECCV* (2016).

[120] OORD, A. V. D., LI, Y., AND VINYALS, O. Representation learning with contrastive predictive coding. *arXiv preprint arXiv:1807.03748* (2018).

[121] PAN, S. J., YANG, Q., ET AL. A survey on transfer learning. *TKDE* (2010).

[122] PARKHI, O. M., VEDALDI, A., ZISSERMAN, A., AND JAWAHAR, C. Cats and dogs. In *CVPR* (2012).

[123] PASZKE, A., GROSS, S., MASSA, F., LERER, A., BRADBURY, J., CHANAN, G., KILLEEN, T., LIN, Z., GIMELSHEIN, N., ANTIGA, L., ET AL. Pytorch: An imperative style, high-performance deep learning library. In *NeurIPS* (2019).

[124] PENNINGTON, J., SOCHER, R., AND MANNING, C. D. Glove: Global vectors for word representation. In *EMNLP* (2014).

[125] PHAM, H., GUAN, M. Y., ZOPH, B., LE, Q. V., AND DEAN, J. Efficient neural architecture search via parameter sharing. In *ICML* (2018).

[126] PINHEIRO, P. O., ALMAHAIRI, A., BENMALECK, R. Y., GOLEMO, F., AND COURVILLE, A. Unsupervised learning of dense visual representations. In *NeurIPS* (2020).

[127] PONT-TUSET, J., PERAZZI, F., CAELLES, S., ARBELÁEZ, P., SORKINE-HORNUNG, A., AND VAN GOOL, L. The 2017 davis challenge on video object segmentation. *arXiv:1704.00675* (2017).

[128] PURUSHWALKAM, S., AND GUPTA, A. Demystifying contrastive self-supervised learning: Invariances, augmentations and dataset biases. In *NeurIPS* (2020).

[129] RAFFEL, C., SHAZEER, N., ROBERTS, A., LEE, K., NARANG, S., MATENA, M., ZHOU, Y., LI, W., AND LIU, P. J. Exploring the limits of transfer learning with a unified text-to-text transformer. *arXiv preprint arXiv:1910.10683* (2019).

[130] RAI, P., AND DAUMÉ III, H. Infinite predictor subspace models for multitask learning. In *AISTATS* (2010).

[131] REAL, E., AGGARWAL, A., HUANG, Y., AND LE, Q. V. Regularized evolution for image classifier architecture search. In *AAAI* (2019).

[132] RONNEBERGER, O., FISCHER, P., AND BROX, T. U-net: Convolutional networks for biomedical image segmentation. In *International Conference on Medical image computing and computer-assisted intervention* (2015).

[133] ROSENBAUM, C., KLINGER, T., AND RIEMER, M. Routing networks: Adaptive selection of non-linear functions for multi-task learning. In *ICLR* (2018).

[134] ROY, A., AND TODOROVIC, S. Monocular depth estimation using neural regression forest. In *CVPR* (2016).

[135] RUDD, E. M., GÜNTHER, M., AND BOULT, T. E. Moon: A mixed objective optimization network for the recognition of facial attributes. In *ECCV* (2016).

[136] RUDER, S., BINGEL, J., AUGENSTEIN, I., AND SØGAARD, A. Latent multi-task architecture learning. In *AAAI* (2019).

[137] SANH, V., WOLF, T., AND RUDER, S. A hierarchical multi-task approach for learning embeddings from semantic tasks. In *AAAI* (2019).

[138] SELVARAJU, R. R., DESAI, K., JOHNSON, J., AND NAIK, N. Casting your model: Learning to localize improves self-supervised representations. In *CVPR* (2020).

[139] SENER, O., AND KOLTUN, V. Multi-task learning as multi-objective optimization. In *NeurIPS* (2018).

[140] SILBERMAN, N., HOIEM, D., KOHLI, P., AND FERGUS, R. Indoor segmentation and support inference from rgbd images. In *ECCV* (2012).

[141] SIMONYAN, K., AND ZISSERMAN, A. Very deep convolutional networks for large-scale image recognition. In *ICLR* (2015).

[142] SINHA, A., CHEN, Z., BADRINARAYANAN, V., AND RABINOVICH, A. Gradient adversarial training of neural networks. *arXiv preprint arXiv:1806.08028* (2018).

[143] SUN, K., XIAO, B., LIU, D., AND WANG, J. Deep high-resolution representation learning for human pose estimation. In *CVPR* (2019).

[144] SUTEU, M., AND GUO, Y. Regularizing deep multi-task networks using orthogonal gradients. *arXiv preprint arXiv:1912.06844* (2019).

[145] TEICHMANN, M., WEBER, M., ZOELLNER, M., CIPOLLA, R., AND URTASUN, R. Multinet: Real-time joint semantic reasoning for autonomous driving. In *IV* (2018).

[146] TIAN, Y., KRISHNAN, D., AND ISOLA, P. Contrastive multiview coding. In *ECCV* (2020).

[147] TIAN, Y., SUN, C., POOLE, B., KRISHNAN, D., SCHMID, C., AND ISOLA, P. What makes for good views for contrastive learning. In *NeurIPS* (2020).

[148] VAN GANSBEKE, W., VANDENHENDE, S., GEORGOULIS, S., PROESMANS, M., AND VAN GOOL, L. Scan: Learning to classify images without labels. In *ECCV* (2020).

[149] VAN GANSBEKE, W., VANDENHENDE, S., GEORGOULIS, S., AND VAN GOOL, L. Revisiting contrastive methods for unsupervised learning of visual representations.

[150] VAN GANSBEKE, W., VANDENHENDE, S., GEORGOULIS, S., AND VAN GOOL, L. Unsupervised semantic segmentation by contrasting object mask proposals. In *ICCV* (2021).

[151] VANDENHENDE, S., DE BRABANDERE, B., NEVEN, D., AND VAN GOOL, L. A three-player gan: generating hard samples to improve classification networks. In *MVA* (2019).

[152] VANDENHENDE, S., GEORGOULIS, S., DE BRABANDERE, B., AND VAN GOOL, L. Branched multi-task networks: Deciding what layers to share. In *BMVC* (2020).

[153] VANDENHENDE, S., GEORGOULIS, S., VAN GANSBEKE, W., PROESMANS, M., DAI, D., AND VAN GOOL, L. Multi-task learning for dense prediction tasks: A survey. *TPAMI* (2021).

[154] VANDENHENDE, S., GEORGOULIS, S., AND VAN GOOL, L. Mti-net: Multi-scale task interaction networks for multi-task learning. In *ECCV* (2020).

[155] VASWANI, A., SHAZEER, N., PARMAR, N., USZKOREIT, J., JONES, L., GOMEZ, A. N., KAISER, Ł., AND POLOSUKHIN, I. Attention is all you need. In *NeurIPS* (2017).

[156] WANG, J., CHENG, Y., AND SCHMIDT FERIS, R. Walk and learn: Facial attribute representation learning from egocentric video and contextual data. In *CVPR* (2016).

[157] WANG, J., SUN, K., CHENG, T., JIANG, B., DENG, C., ZHAO, Y., LIU, D., MU, Y., TAN, M., WANG, X., ET AL. Deep high-resolution representation learning for visual recognition. *TPAMI* (2019).

[158] WANG, P., SHEN, X., LIN, Z., COHEN, S., PRICE, B., AND YUILLE, A. L. Towards unified depth and semantic prediction from a single image. In *CVPR* (2015).

[159] WANG, X., LIU, Z., AND YU, S. X. Unsupervised feature learning by cross-level discrimination between instances and groups. In *CVPR* (2021).

[160] WANG, X., ZHANG, R., SHEN, C., KONG, T., AND LI, L. Dense contrastive learning for self-supervised visual pre-training. In *CVPR* (2020).

[161] WU, Y., KIRILLOV, A., MASSA, F., LO, W.-Y., AND GIRSHICK, R. Detectron2. `https://github.com/facebookresearch/detectron2`, 2019.

[162] WU, Z., XIONG, Y., YU, S. X., AND LIN, D. Unsupervised feature learning via non-parametric instance discrimination. In *CVPR* (2018).

[163] XIAO, J., HAYS, J., EHINGER, K. A., OLIVA, A., AND TORRALBA, A. Sun database: Large-scale scene recognition from abbey to zoo. In *CVPR* (2010).

[164] XIAO, T., REED, C. J., WANG, X., KEUTZER, K., AND DARRELL, T. Region similarity representation learning. *arXiv preprint arXiv:2103.12902* (2021).

[165] XU, D., OUYANG, W., WANG, X., AND SEBE, N. Pad-net: Multi-tasks guided prediction-and-distillation network for simultaneous depth estimation and scene parsing. In *CVPR* (2018).

[166] XU, D., WANG, W., TANG, H., LIU, H., SEBE, N., AND RICCI, E. Structured attention guided convolutional neural fields for monocular depth estimation. In *CVPR* (2018).

[167] XUE, Y., LIAO, X., CARIN, L., AND KRISHNAPURAM, B. Multi-task learning for classification with dirichlet process priors. *JMLR* (2007).

[168] YANG, C., WU, Z., ZHOU, B., AND LIN, S. Instance localization for self-supervised detection pretraining. In *CVPR* (2021).

[169] YANG, Y., AND HOSPEDALES, T. Deep multi-task representation learning: A tensor factorisation approach. *arXiv preprint arXiv:1605.06391* (2016).

[170] YE, M., ZHANG, X., YUEN, P. C., AND CHANG, S.-F. Unsupervised embedding learning via invariant and spreading instance feature. In *CVPR* (2019).

[171] YOSINSKI, J., CLUNE, J., BENGIO, Y., AND LIPSON, H. How transferable are features in deep neural networks? In *NeurIPS* (2014).

[172] YU, F., CHEN, H., WANG, X., XIAN, W., CHEN, Y., LIU, F., MADHAVAN, V., AND DARRELL, T. Bdd100k: A diverse driving dataset for heterogeneous multitask learning. In *CVPR* (2020).

[173] YU, K., TRESP, V., AND SCHWAIGHOFER, A. Learning gaussian processes from multiple tasks. In *ICML* (2005).

[174] YUAN, Y., AND WANG, J. Ocnet: Object context network for scene parsing. *arXiv preprint arXiv:1809.00916* (2018).

[175] ZAMIR, A. R., SAX, A., CHEERLA, N., SURI, R., CAO, Z., MALIK, J., AND GUIBAS, L. J. Robust learning through cross-task consistency. In *CVPR* (2020).

[176] ZAMIR, A. R., SAX, A., SHEN, W., GUIBAS, L. J., MALIK, J., AND SAVARESE, S. Taskonomy: Disentangling task transfer learning. In *CVPR* (2018).

[177] ZHANG, X., AND MAIRE, M. Self-supervised visual representation learning from hierarchical grouping. In *NeurIPS* (2020).

[178] ZHANG, Z., CUI, Z., XU, C., JIE, Z., LI, X., AND YANG, J. Joint task-recursive learning for semantic segmentation and depth estimation. In *ECCV* (2018).

[179] ZHANG, Z., CUI, Z., XU, C., YAN, Y., SEBE, N., AND YANG, J. Pattern-affinitive propagation across depth, surface normal and semantic segmentation. In *CVPR* (2019).

[180] ZHAO, H., SHI, J., QI, X., WANG, X., AND JIA, J. Pyramid scene parsing network. In *CVPR* (2017).

[181] ZHAO, N., WU, Z., LAU, R. W., AND LIN, S. What makes instance discrimination good for transfer learning? In *ICLR* (2020).

[182] ZHAO, X., LI, H., SHEN, X., LIANG, X., AND WU, Y. A modulation module for multi-task learning with applications in image retrieval. In *ECCV* (2018).

[183] ZHOU, B., KHOSLA, A., LAPEDRIZA, A., OLIVA, A., AND TORRALBA, A. Learning deep features for discriminative localization. In *CVPR* (2016).

[184] ZHOU, B., LAPEDRIZA, A., KHOSLA, A., OLIVA, A., AND TORRALBA, A. Places: A 10 million image database for scene recognition. *T-PAMI* (2017).

[185] ZHOU, J., CHEN, J., AND YE, J. Clustered multi-task learning via alternating structure optimization. In *NeurIPS* (2011).

[186] ZHOU, L., CUI, Z., XU, C., ZHANG, Z., WANG, C., ZHANG, T., AND YANG, J. Pattern-structure diffusion for multi-task learning. In *CVPR* (2020).

[187] ZOPH, B., AND LE, Q. V. Neural architecture search with reinforcement learning. In *ICLR* (2017).

Curriculum

Simon Vandenhende was born in Roeselare, Belgium in 1995. He received his bachelor's and master's degree in Electrical Engineering from KU Leuven. During his master's, he worked as an intern at the SI2 lab, NCTU, Taiwan. Later on, Simon developed a strong interest in machine learning and computer vision. This led him to join the VISICS computer vision lab at KU Leuven in 2017 to pursue a PhD under the supervision of Prof. Luc van Gool. His research focuses on multi-task learning and self-supervised representation learning. In 2019, Simon won a best paper award at the International Conference on Machine Vision Applications. He also co-organized workshops on multi-modal learning and multi-task learning at ECCV'20 and ICCV'21 respectively. In Fall 2021, he participated in a research internship at Facebook AI Research (FAIR).

List of Publications

Articles in international peer reviewed conferences or journals

(*) denotes equal contribution.

- Van Gansbeke, W.*, **Vandenhende, S.***, Georgoulis, S., & Van Gool, L. (2021). Revisiting Contrastive Methods for Unsupervised Learning of Visual Representations. In Conference on Neural Information Processing Systems. [149]
- Van Gansbeke, W.*, **Vandenhende, S.***, Georgoulis, S., & Van Gool, L. (2021). Unsupervised semantic segmentation by contrasting object mask proposals. In International Conference on Computer Vision. [150]
- **Vandenhende, S.**, Georgoulis, S., Van Gansbeke, W., Proesmans, M., Dai, D., & Van Gool, L. (2021). Multi-task learning for dense prediction tasks: A Survey. IEEE Transactions on Pattern Analysis and Machine Intelligence. [153]
- **Vandenhende, S.**, Georgoulis, S., De Brabandere, B., & Van Gool, L. (2020) Branched Multi-Task Networks: Deciding What Layers To Share. In British Machine Vision Conference. [152]
- **Vandenhende, S.**, Georgoulis, S., & Van Gool, L. (2020). Mti-net: Multi-scale task interaction networks for multi-task learning. In European Conference on Computer Vision. [154]
- Van Gansbeke, W.*, **Vandenhende, S.***, Georgoulis, S., Proesmans, M., & Van Gool, L. (2020). Scan: Learning to classify images without labels. In European Conference on Computer Vision. [148]
- Deruyttere, T., **Vandenhende, S.**, Grujicic, D., Van Gool, L., & Moens, M. F. (2019). Talk2Car: Taking Control of Your Self-Driving Car. In Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing

and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP). [35]

- **Vandenhende, S.**, De Brabandere, B., Neven, D., & Van Gool, L. (2019). A three-player GAN: generating hard samples to improve classification networks. In 2019 16th International Conference on Machine Vision Applications. [151]

Articles in international peer reviewed workshops

- Deruyttere, T., **Vandenhende, S.**, Grujicic, D., Liu, Y., Van Gool, L., Blaschko, M., ... & Moens, M. F. (2020, August). Commands 4 Autonomous Vehicles (C4AV) Workshop Summary. In European Conference on Computer Vision. [34]

Workshops

- Commands For Autonomous Vehicles Workshop (C4AV), ECCV, August, 2020.
- Workshop on Multi-Task Learning in Computer Vision (DeepMTL), ICCV, October, 2021.

Awards

- Best paper award at the International Conference on Machine Vision Applications, Tokyo, May, 2019.



FACULTY OF ENGINEERING SCIENCE
DEPARTMENT OF ELECTRICAL ENGINEERING
ESAT - PSI
Kasteelpark Arenberg 10 - box 2441
B-3001 Leuven
simon.vandenhende@esat.kuleuven.be
https://github.com/SimonVandenhende