



# Optimization of deep multi-task networks

Lucas Pascal

## ► To cite this version:

Lucas Pascal. Optimization of deep multi-task networks. Neural and Evolutionary Computing [cs.NE]. Sorbonne Université, 2021. English. NNT : 2021SORUS535 . tel-03783509

**HAL Id: tel-03783509**

**<https://theses.hal.science/tel-03783509>**

Submitted on 22 Sep 2022

**HAL** is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

SORBONNE UNIVERSITY  
DOCTORAL SCHOOL EDITE  
DATA SCIENCE

# P H D T H E S I S

to obtain the title of

**PhD of Science**

of EURECOM - Sophia Antipolis

**Specialty : DATA SCIENCE**

Defended by

Lucas PASCAL

## Optimization of Deep Multi-Task Networks

Thesis Advisor: Maria A. ZULUAGA

prepared at EURECOM Sophia Antipolis

defended on 08/11/2021

### **Jury :**

<i>Reviewers :</i>	Pablo ÁRBELAEZ	-	Universidad de los Andes
	Marcel WORRING	-	University of Amsterdam
<i>Examiners :</i>	François BREMOND	-	INRIA
	Ender KONUKOGLU	-	ETH Zurich
	Pietro MICHIARDI	-	Eurecom
	Maria A. ZULUAGA	-	Eurecom
<i>Invited :</i>	Xavier BOST	-	Orkis
	Benoit HUET	-	Median Technologies



## Acknowledgments

During this PhD, I've been surrounded by many great people, all of whom have contributed in their own way to my work. I would like to express my gratitude to them in the following lines.

I would first like to thank my two successive supervisors Benoit Huet and Maria A. Zuluaga. Benoit entrusted me to start this PhD, and kept providing me with valuable advice all along this journey, even after his departure. Maria drove me forward as a researcher, and I learnt a lot from her rigor. Her kindness and encouragements in periods of doubt have been an incredible support to me.

I am also very grateful to Xavier, my Orkis supervisor for his kindness during these almost four years in the company. Working with him has been both enjoyable and enriching to me.

I have then to thank all the PhDs and researchers who interacted with me in Eu-recom during these years, and created a favorable research environment for my growth.

Many thanks to everyone in Orkis. It's been a pleasure to work in this motivated, lively and friendly team. All the moments in the company were pleasant ones to me, and I feel proud to have contributed to its development.

I'd finally like to thank my family and all my friends for their continuous presence and support. I feel blessed to be surrounded by such wonderful people, and this positive environment has always been a key factor in my progress.



# Abstract

Multi-Task Learning (MTL) is a learning paradigm involving the joint optimization of parameters with respect to multiple tasks. By learning multiple related tasks, a learner receives more complete and complementary information on the input domain from which the tasks are issued. This allows to gain better understanding of the domain by building a more accurate set of assumptions of it, what is denoted as the *inductive bias*, thus leading to more robust features and better generalization performance.

Thanks to these features, Multi-Task Learning has become highly attractive for deep networks. Given their important data requirements, MTL offers the possibility to learn more generalizable features from the often limited data at disposal. However, in practice, the broader use of MTL is hindered by the lack of consistent performance gains observed by deep multi-task networks. It is often the case that deep MTL networks suffer from performance degradation caused by *task interference*. This phenomenon results from destructive interactions between the back-propagated gradients of the different tasks, and prevents the network from strengthening its inductive bias.

This thesis addresses the problem of task interference in MTL to improve the generalization capabilities of deep networks. To this end, it introduces novel optimization techniques for multi-task learners, which take inspiration from existing partitioning methods. These methods tackle task interference by allowing the tasks to specify their own usage of network neurons. Differently from standard MTL optimization schemes, which aggregate all task specific objectives under a unique multi-task objective optimized using gradient descent methods, partitioning methods use independent gradient descent steps that are alternated along the different task-specific objectives.

In a first contribution, a new dynamic parameter sharing scheme is proposed, which reduces task interference while strengthening the inductive bias by maximizing the contribution of every task in the learning of every parameter. This is done with a lightweight discrete partitioning update scheme, inspired from dropout and based on random updates. The proposed method, Maximum Roaming, achieved better generalization performance over different state-of-the-art multi-task baselines, across varied experimental settings.

In a second contribution, this thesis studies the difference between the optimization schemes of standard MTL and partitioning methods, which surprisingly has not been studied in detail before. To this end, it presents a convergence analysis, along with

a comparative study. From this inspiration, in a third contribution is introduced a novel optimization scheme, which pushes further the separation of task specific objectives by specifying task-specific momentum mechanisms, aiming for a more stochastic optimization. A fourth contribution proposes a task grouping strategy, in order to freely compromise between the improved generalization performances of this novel optimization scheme, and the computational efficiency of the standard optimization scheme used in most related works.

In a final contribution, the developed methods are used in a concrete real world application, involving glaucoma diagnosis from retinal fundus images. A MTL approach is proposed, to create superior generalization performance from the few data at disposal. Although task interference is observable in the experimental results of the multi-task baselines, the proposed methods efficiently mitigate it, and are able to improve the generalization performance of the pipeline compared to single task models.

# Résumé

L'apprentissage multi-tâches est un paradigme d'apprentissage qui consiste à optimiser des paramètres par rapport à plusieurs tâches simultanément. En apprenant plusieurs tâches liées, un modèle d'apprentissage dispose d'un ensemble d'informations plus complet concernant le domaine d'entrée dont les tâches sont issues. Cela lui permet de mieux appréhender ce domaine, en construisant un ensemble d'hypothèses plus précis, appelé *biais inductif*, menant ainsi à de meilleures représentations et une meilleure généralisation.

Ces avantages ont rendu l'apprentissage multi-tâches très attractif pour les réseaux de neurones profonds. Étant données les contraintes qu'ils imposent en termes de quantités de données, l'apprentissage multi-tâches offre la possibilité d'apprendre des représentations plus robustes à partir des données souvent limitées à disposition. Cependant, en pratique, la systématisation de méthodes multi-tâches est limitée par le manque de régularité dans les gains de performance obtenus par les réseaux multi-tâches. Il arrive au contraire que ces réseaux multi-tâches subissent une perte de performance causée par des *interférences de tâches*. Ce phénomène résulte d'interactions destructives entre les gradients rétro-propagés par les différentes tâches, et empêche les réseaux de renforcer leur biais inductif.

Cette thèse traite du problème d'interférences de tâches en apprentissage multi-tâches, afin d'améliorer les capacités de généralisation des réseaux de neurones profonds. Elle introduit ainsi de nouvelles techniques d'optimisation de réseaux multi-tâches, en s'inspirant de méthodes de partitionnement existantes. Ces méthodes réduisent les phénomènes d'interférences en permettant aux différentes tâches de spécifier leur propre usage des neurones du réseau. Contrairement aux méthodes usuelles d'apprentissage multi-tâches, qui accumulent les objectifs propres aux différentes tâches sous un unique objectif multi-tâches, les méthodes de partitionnement alternent les descentes de gradient sur les objectifs respectifs des différentes tâches.

Dans une première contribution, un nouveau schéma dynamique de partage des paramètres est proposé. Celui-ci réduit les phénomènes d'interférences et renforce le biais inductif en garantissant la contribution de chaque tâche dans l'apprentissage de chaque paramètre partagé. Pour cela est utilisé un schéma de partitionnement dynamique binaire et peu coûteux, inspiré du dropout et basé sur des mises à jour aléatoires. La méthode proposée, Maximum Roaming, atteint de meilleures performances de généralisation par rapport à différentes approches multi-tâches existant dans l'état de l'art, dans des contextes expérimentaux variés.

Dans une seconde contribution, cette thèse étudie la différence entre l'optimisation



utilisée par les méthodes usuelles d'apprentissage multi-tâches, et celle utilisée par les méthodes de partitionnement, qui n'a étonnement jamais été étudiée en détails. Pour cela, une étude de convergence et une étude expérimentale comparative de ces différents schémas d'optimisation est présentée. Dans la continuité de ce travail, une troisième contribution propose un nouveau schéma d'optimisation, qui pousse plus loin la séparation entre les objectifs propres aux différentes tâches en spécifiant des mécanismes de momentum eux aussi propres aux différentes tâches, dans le but de favoriser une optimisation plus stochastique. Une quatrième contribution propose une stratégie de groupement de tâches, afin d'établir librement différents compromis entre les performances de généralisation accrues de ce nouveau schéma d'optimisation et la vitesse de calcul du schéma d'optimisation utilisé dans la plupart des travaux existants en apprentissage multi-tâches.

Dans une dernière contribution, les méthodes développées dans cette thèse sont utilisées dans une application concrète, à savoir le diagnostic automatique de glaucome à partir d'images de fundus rétinien. Une approche multi-tâches est proposée afin d'obtenir de meilleures performances de généralisation à partir du peu de données annotées à disposition. Bien que des phénomènes d'interférences soient empiriquement observables sur des méthodes d'apprentissage multi-tâches standards, les méthodes proposées les réduisent efficacement et sont capables d'améliorer les performances de généralisation du réseau par rapport à des équivalents mono-tâches.

# Contents

<b>1</b>	<b>Introduction</b>	<b>1</b>
1.1	Using related tasks to strengthen the inductive bias . . . . .	3
1.1.1	Transfer Learning . . . . .	3
1.1.2	Continual Learning . . . . .	4
1.1.3	Multi-Task Learning . . . . .	5
1.2	Deep Multi-Task Learning . . . . .	5
1.2.1	Parameter Sharing . . . . .	6
1.2.2	Optimization of the shared network . . . . .	6
1.2.3	Task interference in Deep MTL . . . . .	7
1.3	Contributions . . . . .	8
1.4	Publications . . . . .	9
1.5	Thesis Outline . . . . .	10
<b>2</b>	<b>Literature Review</b>	<b>11</b>
2.1	Tackling task interference in Deep Multi-Task Learning . . . . .	11
2.1.1	Task affinities in MTL . . . . .	11
2.1.2	Network Architectures in MTL . . . . .	13
2.1.3	Multi-task Optimization . . . . .	14
2.2	Related works in MTL optimization . . . . .	15
2.2.1	Loss weighting . . . . .	15
2.2.2	Multi-Objective Optimization . . . . .	16
2.2.3	Gradient editing . . . . .	17
2.2.4	Parameter partitioning . . . . .	18
<b>3</b>	<b>Strengthening the inductive bias with a dynamic parameter partitioning</b>	<b>21</b>
3.1	Motivation . . . . .	21
3.2	Preliminaries . . . . .	23
3.2.1	Parameter Partitioning . . . . .	24
3.2.2	Parameter Partitioning Initialization . . . . .	24
3.3	Maximum Roaming Multi-Task Learning . . . . .	25
3.4	Experimental Results . . . . .	27
3.4.1	Datasets . . . . .	27
3.4.2	Baselines . . . . .	28
3.4.3	Facial Attributes Detection . . . . .	29
3.4.4	Scene Understanding . . . . .	33
3.5	Discussion . . . . .	37
3.6	Appendix . . . . .	38

<b>4</b>	<b>Separating task-specific objectives for a better optimization</b>	<b>41</b>
4.1	Motivation . . . . .	41
4.2	Alternate and independent optimization of task-specific objective functions . . . . .	43
4.2.1	Standard MTL optimization with aggregated loss . . . . .	43
4.2.2	Alternate and independent optimization of task-specific objective functions for SGD . . . . .	44
4.2.3	Alternate and independent optimization of task-specific objective functions for moving-average based optimizers . . . . .	45
4.2.4	Mitigating computational costs through task grouping . . . . .	46
4.3	Experiments and results . . . . .	47
4.3.1	Scene understanding on NYUv2 . . . . .	47
4.3.2	Multi-class segmentation on Cityscapes . . . . .	49
4.3.3	Multi-attribute segmentation on Celeb-A . . . . .	50
4.3.4	Covered distance . . . . .	52
4.4	Discussion . . . . .	53
4.5	Appendix . . . . .	54
<b>5</b>	<b>Glaucoma Diagnosis from Retinal Fundus Imaging through MTL</b>	<b>61</b>
5.1	Motivation . . . . .	61
5.1.1	Deep multi-task networks for automated glaucoma diagnosing	61
5.1.2	The Retinal Fundus Imaging challenge (REFUGE) . . . . .	63
5.2	Related work . . . . .	64
5.3	Method . . . . .	64
5.3.1	Pipeline description . . . . .	65
5.3.2	Losses and metrics . . . . .	67
5.3.3	Optimization . . . . .	68
5.4	Experiments and Results . . . . .	68
5.4.1	Experimental details . . . . .	68
5.4.2	Experimental Results . . . . .	69
5.4.3	Combination with Transfer Learning . . . . .	73
5.5	Discussion . . . . .	76
<b>6</b>	<b>Conclusion</b>	<b>79</b>
6.1	Summary of contributions . . . . .	79
6.2	Open perspectives . . . . .	81
	<b>Appendices</b>	<b>83</b>
<b>A</b>	<b>Semantic and Visual Similarities for Efficient Knowledge Transfer in CNN Training</b>	<b>85</b>
A.1	Introduction . . . . .	85
A.2	Related Works . . . . .	87
A.2.1	Datasets and architectures . . . . .	87

---

A.2.2	Transfer Learning process . . . . .	87
A.2.3	Semantic similarity between textual content . . . . .	88
A.3	Similarity-based knowledge transfer . . . . .	89
A.3.1	Similarity measures . . . . .	89
A.3.2	Initialization . . . . .	90
A.4	Experiments and Results . . . . .	90
A.4.1	Implementation Details . . . . .	90
A.4.2	Dataset . . . . .	91
A.4.3	Similarities and Initialization . . . . .	91
A.4.4	Neighboring optimization . . . . .	92
A.4.5	Data reduction study . . . . .	94
A.5	Conclusion and Perspectives . . . . .	95
<b>Bibliography</b>		<b>97</b>



# Introduction

---

## Contents

<b>1.1</b>	<b>Using related tasks to strengthen the inductive bias . . . .</b>	<b>3</b>
1.1.1	Transfer Learning . . . . .	3
1.1.2	Continual Learning . . . . .	4
1.1.3	Multi-Task Learning . . . . .	5
<b>1.2</b>	<b>Deep Multi-Task Learning . . . . .</b>	<b>5</b>
1.2.1	Parameter Sharing . . . . .	6
1.2.2	Optimization of the shared network . . . . .	6
1.2.3	Task interference in Deep MTL . . . . .	7
<b>1.3</b>	<b>Contributions . . . . .</b>	<b>8</b>
<b>1.4</b>	<b>Publications . . . . .</b>	<b>9</b>
<b>1.5</b>	<b>Thesis Outline . . . . .</b>	<b>10</b>

---

Deep neural networks (DNN) are known for being over-parameterized models, containing more parameters than the number of available training samples. This makes them prone to overfitting, leading to poor generalization on new data. It is thus crucial to inject DNNs with massive latent information about the input domain from which the samples are extracted, so that the DNNs can build a relevant set of assumptions about these domain rules. For example, when working with natural images, to generalize correctly a learner should learn the invariance of certain entities to pose and lighting. These domain rules are not explicitly provided in data labels, but can be deduced by the learner when it experiences label invariance for images of different pose and lighting. The learner will then refine pose and lighting invariant features, which will generalize better on new data. Reciprocally, when learning from low populated and poorly varied datasets, the lack of latent information on the input domain can lead the learner to deduce wrong assumptions, which will lead to poor generalization, i.e. overfitting. This set of assumptions made by a learner on an input domain is generally called *inductive bias* in machine learning [Baxter 2000].

One first way to strengthen the inductive bias of a learner is to feed it with vast amounts of data: this is how the massive increase of online publicly available annotated data led to an impressive breakthrough of deep networks these recent years [Deng *et al.* 2009], and established them as a standard in most computer vision tasks [Krizhevsky *et al.* 2012]. However, creating massive sets of annotated data for

a given task is not affordable in the general case: First, collecting such amount of data is often impossible in many domains, like medical imaging [Tajbakhsh *et al.* 2020] in which data privacy makes it really hard for medical centers to share data, or for specific tasks in which data doesn't exist in large amounts. Second, data labelling is often a tedious and costly work, which cannot reasonably be applied to millions of samples. For example, annotations for tasks like video captioning or image segmentation can require a considerable time for just one sample [Dang *et al.* 2021]. Hence, other solutions have to be adopted to improve the generalization performance of DNNs.

Data augmentation, dropout and norm-regularization are common and widely used techniques to improve DNNs generalization [Tajbakhsh *et al.* 2020]. Data augmentation refers to techniques used to increase the amount of data by adding slightly modified copies of the available data or newly created synthetic data using the existing data as a starting point. These include geometric transformations, noise addition and color shifts, among others [Shorten & Khoshgoftaar 2019]. Dropout [Srivastava *et al.* 2014] addresses the generalization problem by explicitly trying to reduce overfitting of DNNs. To this end, network units are randomly switched off (or dropped out) during training to encourage the network to rely on as many possible feature combinations, thus preventing units to co-adapt too much. At inference time, the predictions from the different "thinned" networks are averaged together. Finally, norm-regularization techniques penalize the norm ( $L1$  or  $L2$ ) of parameters during learning, which is shown to reduce erratic behaviors between interpolated samples, therefore improving generalization.

Another important resource to improve generalization when training a DNN on a given task is the usage of other related tasks to introduce a positive inductive bias. Since different related tasks may provide different and complementary information on their input domain, using multiple related tasks to train a model may introduce a more robust inductive bias, and lead the model to better generalization [Caruana 1997, Baxter 2000]. This is why humans perform so well on any new vision task: new tasks are not learned from scratch, but instead reuse all related knowledge on the visual domain, and quickly adapt to this new task [Thrun & Pratt 1998]. Multi-task learning is a machine learning sub-domain, which formalizes this concept and aims to develop learning-based methods that can efficiently exploit multiple tasks in their training process. This thesis focuses on deep Multi-Task Learning, which aims at jointly optimizing a DNN with respect to multiple tasks, in order to strengthen its inductive bias.

This chapter provides a context to the work presented in the thesis. It first details the different existing methods to introduce an inductive bias in the learning process of a deep neural network from the interactions of related tasks. In a second part, more attention is given to Multi-Task Learning. Its application to deep networks (i.e. deep Multi-Task Learning) is more thoroughly detailed and task interference,

the main challenge in MTL is presented. This thesis is devoted to improve task interference in MTL methods. The chapter concludes by summarizing the main contributions of this thesis, listing related publications, and by presenting the outline of this manuscript.

## 1.1 Using related tasks to strengthen the inductive bias

The usage of multiple related tasks to create robust inductive bias is mainly studied in three domains, which are *Transfer Learning*, *Multi-Task Learning* and *Continual Learning*.

### 1.1.1 Transfer Learning

In *Transfer Learning* [Pratt 1993, Zamir *et al.* 2018], a model previously trained on a source task is used to learn a related but different target task from the same input domain. Using the source task as a proxy for the target task instead of training a model *from scratch* directly on the target task introduces into the learning process of the target task a set of assumptions on the input domain (i.e. inductive bias) learned on the source task. This complementary knowledge on the input domain might not have been acquired as such by training on the target task alone, and should help the model to learn more generalizable features for the target task. This is particularly useful in cases where annotated data is lacking for the target task: the already robust inductive bias learned on the source task allows the network to quickly and efficiently adapt its robust features to the new task with only few samples.

Transfer Learning for neural networks has been firstly introduced in [Hinton & Salakhutdinov 2006]. It was then studied in more details in works like [Yosinski *et al.* 2014] and [Chu *et al.* 2016]. In [Yosinski *et al.* 2014] experiments transfers of different network depths, and notice that transfers can be negatively affected when transferring only portions of pre-trained networks, which could break some co-adaptations between convolutional filters of different depth. They also notice that deeper layers tend to contain more specialized features, which do not transfer as well as shallow layers. Authors in [Chu *et al.* 2016] conduce their study depending on different target dataset sizes, and conclude that the performance gains of Transfer Learning compared to models trained from scratch is all the more so high when the target dataset is small. The transfer affinities between different related computer vision tasks have been thoroughly studied in works like [Zamir *et al.* 2018], which provides a large transfer affinity graph between 26 tasks. Some works [Wang *et al.* 2017, Gonzales Zuniga *et al.* 2018] also showed that the performance gains of Transfer Learning can be further increased by augmenting the pre-trained network with more weights: this is done in [Wang *et al.* 2017] by deepening and widening the pre-trained network with new units. Similarly, in [Gonzales Zuniga *et al.* 2018], residual units are added to the transferred network, and trained under a specific scheme, to allow the network to better compensate the



shift between the source and target task.

Transfer learning has become very popular in deep learning, in which data requirements make it difficult to train networks from scratch. It has shown as particularly beneficial in computer vision, inside of which the different tasks generally transfer very well. In practice, the pre-trained layers in convolutional networks transfer so well between vision tasks that Transfer Learning has also shown as beneficial between different domains: for example in medical imaging [Tajbakhsh *et al.* 2020], for which models pre-trained on natural images (such as ImageNet [Deng *et al.* 2009]) are often used to compensate for the lack of annotated data. However, Transfer Learning is a one-sided interaction between tasks that does not aim to maintain performance on the source task.

### 1.1.2 Continual Learning

In *Continual Learning* [Parisi *et al.* 2019], a learner continuously learns new incoming tasks without forgetting previous ones, i.e. avoid *catastrophic forgetting*, while data from previous tasks might not be available anymore when learning a new task. Unlike Transfer Learning, by retaining the knowledge progressively acquired through different encountered new tasks, the model gets its inductive bias continuously strengthened. This is analogous to how humans progressively learn from birth, by continuously receiving new information, and processing it relatively to the ever-growing knowledge already acquired on their surrounding world (i.e. the input domain).

To avoid catastrophic forgetting, Continual Learning works have to ensure that the knowledge acquired from previous task gets preserved when learning a new one. In works like [Kirkpatrick *et al.* 2017, Zenke *et al.* 2017], this is done through regularization methods: the change on neurons evaluated as relevant with respect to previous tasks gets penalized in a loss term. This relevance is computed as a posterior distribution in [Kirkpatrick *et al.* 2017], while it is computed from entire learning trajectories in the parameter space in [Zenke *et al.* 2017]. In [Lopez-Paz & Ranzato 2017, Chaudhry *et al.* 2019], memories of past gradients from previous tasks are retained, and each learning step for a new task is conducted in order to avoid conflicts with these past gradients. The intuition is to make the network evolve in directions which do not destroy what was learned from previous tasks. Other works instead act on the network architecture [Rusu *et al.* 2016, Mallya *et al.* 2018, Mancini *et al.* 2018], and create task-specific network branches for every new incoming task. The learning is then conducted only in the task-specific portions of the network, while the shared parts are fixed. In [Mallya *et al.* 2018] and [Mancini *et al.* 2018] this is done with trainable task-specific masks that modulate each neuron. Instead, in [Rusu *et al.* 2016] entire new networks are defined for each incoming task. Lateral connections allow to connect with the networks from other tasks.

In most of the Continual Learning literature [Parisi *et al.* 2019], the different tasks are always incoming sequentially, i.e. a new task comes in when the learning of the previous one is finished. Similarly to Transfer Learning, this implies that the training for a new task always start with a network containing a well trained set of assumptions on the input domain. As a result, the case where all tasks income together and have to be learned jointly is not covered. This setting provides at once all latent information on the input domain instead of progressively delivering it through incoming tasks. This simultaneous learning of multiple tasks is also an ability of the human learning system, which has to deal with new incoming tasks both sequentially and jointly. This thesis is devoted to this simultaneous learning of multiple tasks, denoted as Multi-Task Learning.

### 1.1.3 Multi-Task Learning

In *Multi-Task Learning* [Caruana 1997], a learner is trained jointly on multiple tasks from a same input domain. During a unique training phase, the learner is thus exposed to all the available information of the input domain. All the tasks are used to create inductive bias for each other. On top of the strengthened inductive bias, another benefit of MTL is its computational efficiency. In terms of training time, memory and computational power, training a unique network jointly on multiple tasks is generally way more efficient than learning these tasks separately. However the joint optimization of multiple tasks is particularly challenging for deep networks, and is currently an active research topic [Ruder 2017, Crawshaw 2020, Vandenhende *et al.* 2021].

Multi-Task Learning is a widely studied topic in the machine learning field [Zhang & Yang 2021]. More specifically, deep MTL (i.e. MTL with deep networks) has been applied in different domains, such as computer vision (CV) [Ruder 2017, Crawshaw 2020, Vandenhende *et al.* 2021] and Natural Language Processing (NLP) [Collobert & Weston 2008, Liu *et al.* 2015a, Crawshaw 2020]. Most of these works proceed under supervised learning settings, although more recently MTL has been applied with Reinforcement Learning (RL) [Pinto & Gupta 2017, Yu *et al.* 2020, Crawshaw 2020]. In this work, the study is focused on supervised computer vision tasks, which has been the most investigated domain so far [Ruder 2017, Crawshaw 2020, Vandenhende *et al.* 2021].

## 1.2 Deep Multi-Task Learning

This section provides an overview of deep Multi-Task Learning for computer vision. It first introduces the standard application of Multi-Task Learning to Deep Convolutional Networks, detailing how parameters are shared and how models are optimized. Then the performance degradation faced by such networks compared to single task approaches, namely *task interference*, is discussed.

### 1.2.1 Parameter Sharing

Multi-Task Learning uses a unique model for the predictions of multiple tasks. Figure 1.1 depicts the structure typically followed by deep MTL networks. There is a common trunk shared by the different tasks that contains most of the network’s complexity, and task specific prediction heads, made of a few layers [Ruder 2017, Crawshaw 2020]. The inductive bias is generated by the shared parameters in the common trunk, which are jointly trained with respect to every task. The shared trunk, learning from all the tasks, has access to all the latent information on the input domain, and can thus learn more robust features, that the task-specific heads can freely use for their predictions.

Parameter sharing can be of two types: *hard sharing* and *soft sharing*, although some recent works developed hybrid methods for Multi-Task Learning involving both types of sharing [Vandenhende *et al.* 2021].

- *Hard sharing* [Chen *et al.* 2018, Strezoski *et al.* 2019b] consists in using a same set of parameters for every task. This unique set of parameters is trained with respect to every task jointly. It is a particularly computationally efficient approach for deep MTL, since the heaviest part of the network (i.e. the shared trunk) keeps its size fixed regardless of the number of tasks. Adding a new task only incurs a supplementary prediction head, of negligible size compared to the trunk.
- In *soft sharing* [Misra *et al.* 2016, Gao *et al.* 2019], each task uses its own set of weights, but a constraint is added on the distance between all the identical shared weights, thus encouraging them to remain close to each other. This constraint generally takes the form of a L1-norm, L2-norm or the trace norm. The model size thus increases proportionally to the number of tasks, which makes soft sharing strategies difficult to use with heavy models like deep neural networks when the number of tasks increases.

As most existing deep MTL works [Ruder 2017, Crawshaw 2020, Vandenhende *et al.* 2021], this thesis focuses on hard parameter sharing strategies.

### 1.2.2 Optimization of the shared network

Training of a MTL network or model requires that each task is associated to a differentiable *objective function* or *loss function*, to be minimized. To handle the joint optimization of multiple tasks, the standard practice is to average all the task-specific objective functions, forming a unique aggregated objective function (*multi-task objective*), such as:

$$\mathcal{L} = \sum_{i=1}^N w_i \cdot \mathcal{L}_i, \quad (1.1)$$

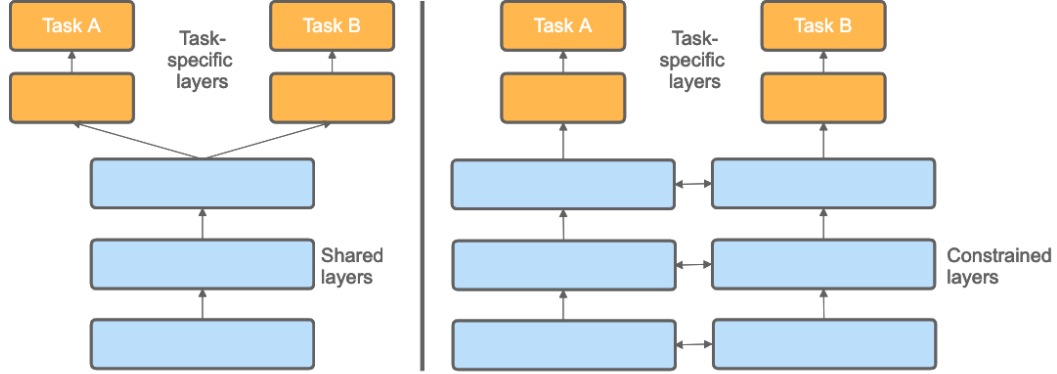


Figure 1.1: **(Right)** Hard parameter sharing. **(Left)** Soft parameter sharing.

with  $\mathcal{L}$  the aggregated multi-task objective function,  $\mathcal{L}_i$  the objective function for task  $i$  and  $w_i$  its associated weight in the averaged sum. The model is trained to minimize this aggregated objective function [Caruana 1997] through gradient descent methods. An example is provided in Figure 1.2, with the representation of both task-specific objective functions and aggregated multi-task objective function.

MTL optimization holds similarities with multi-label classification, where the loss contributions of each class are averaged. The difference between the two consists in that in MTL the objective or loss function may be different for each task. Instead, in multi-label classification the loss function is the same across all classes.

### 1.2.3 Task interference in Deep MTL

Deep networks present high-dimensional, complex and non-convex loss landscapes [Li *et al.* 2018], for which few insights exist at current time, making their optimization a difficult process. Unfortunately, the generalization benefits that could bring MTL approaches are highly dependent on how well the optimization goes. In practice, current MTL approaches are not always able to improve the generalization of single-task models, and can even deteriorate it, suggesting that the joint optimization of parameters by gradient descent methods with respect to multiple non-convex objective functions is a challenging bottleneck to achieve performance gains [Kendall *et al.* 2018]. As shown in Figure 1.2, in non-convex settings, the gradient information obtained from a location in the loss landscape can be misleading for long term improvement. It is therefore not possible to define an optimal and unique way to use the gradients from the different objective functions (which likely point in different directions) without any insight on the shape of the loss landscape. This performance degradation phenomenon observed in Multi-Task Learning is generally called *task interference* or *negative interference* in the literature [Maninis *et al.* 2019, Strezoski *et al.* 2019a], and it is an important hurdle to the spread of multi-task strategies in deep learning systems.

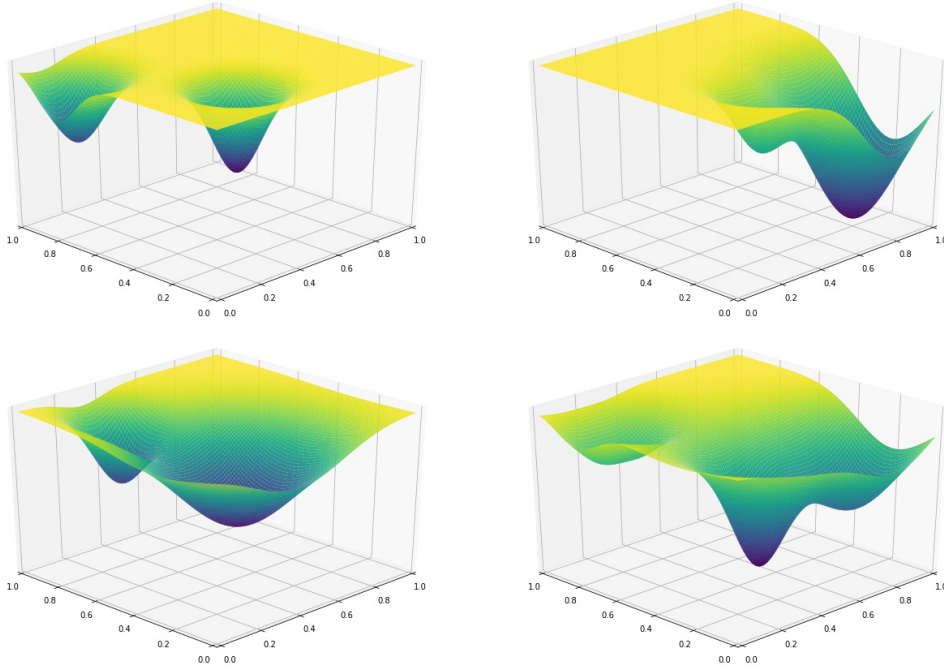


Figure 1.2: **(upper-left)**, **(upper-right)** and **(lower-left)** represent three possible 2-dimensional non-convex objective functions, with **(lower-right)** their sum. From a randomly sampled initialization point for the multi-task optimization, with only local gradient information from each task, it is impossible to guarantee convergence into the multi-task optimum, nor is it to judge if a task gradient is beneficial or detrimental. Increasing the convergence speed or precision would likely not higher the chances of visiting the convex region around the optimum. One might instead prefer to play on the stochasticity of the optimization.

This thesis aims at improving the generalization capabilities of MTL frameworks by improving their optimization. More specifically, it focuses on developing strategies for parameter sharing and optimization that can mitigate task interference, to allow these frameworks better exploit the multiple tasks and build a better inductive bias. Therefore, particular attention is paid to the generalization of the proposed methods to different networks, contexts, and settings.

### 1.3 Contributions

The contributions of this thesis are :

- A method maximising the regularization brought by the plurality of tasks, while reducing task interference. To this end a partitioning strategy is applied to the shared parameters with respect to the different tasks, to reduce the

average number of tasks optimizing a same parameter. The partitioning evolves through the learning process, for each task, thus it is successively involved in the training of every single parameter (Chapter 3).

- A convergence analysis of the specific optimization scheme used in partitioning methods, including the partitioning method proposed in Chapter 3. It optimizes separately and successively the different task-specific objective functions, instead of aggregating them in a single objective to optimize. An experimental comparison with the standard MTL optimization scheme is also provided (Chapter 4).
- A novel optimization scheme inspired by the one used in partitioning methods. It provides more independence to the tasks by defining task-specific momentum mechanisms. This new scheme consistently improves the networks generalization under different experimental settings (Chapter 4).
- A task grouping strategy, which offers a simple and efficient trade-off between the computational overhead and the generalization benefits of the proposed methods (Chapter 4).
- A full application over a real world medical imaging challenge, centered on glaucoma diagnosis from fundus images (Chapter 5), to evidence the benefits brought by the proposed solutions out of an academic context. In particular, they prove to be an efficient solution to increase the multi-task performances in low data settings, and combine well with Transfer Learning.

As previously stated, this thesis focuses on computer vision tasks. However, the main differences in MTL methods across different application domains (e.g. NLP or CV) come from the way that domain-specific architectures are exploited. MTL weight sharing and optimization strategies are essentially domain agnostic. Therefore, the considerations and contributions of this thesis could be easily applied to other domains beyond CV tasks.

## 1.4 Publications

1. **Lucas Pascal**, Xavier Bost and Benoit Huet. *Semantic and Visual Similarities for Efficient Knowledge Transfer in CNN Training*. 2019 International Conference on Content-Based Multimedia Indexing (CBMI), pages 1-6, 2019.
2. **Lucas Pascal**, Benoit Huet, Xavier Bost and Maria A. Zuluaga. *Detection, Segmentation and localization using a single model in Glaucoma detection from color fundus images*. Presented in MICCAI, 2020.
3. **Lucas Pascal**, Pietro Michiardi, Xavier Bost, Benoit Huet and Maria A. Zuluaga. *Maximum Roaming Mutli-Task Learning*. Proceedings of the AAAI Conference on Artificial Intelligence, pages 9331-9341, 2021.

4. **Lucas Pascal**, Oscar J. Perdomo, Xavier Bost, Benoit Huet, Sebastian Otálora and Maria A. Zuluaga. *Multi-task deep learning for Glaucoma detection from color fundus images*. Submitted.
5. **Lucas Pascal**, Pietro Michiardi, Xavier Bost, Benoit Huet and Maria A. Zuluaga. *Optimization Strategies in Multi-Task Learning: Averaged or Independent Losses?* To be submitted.

## 1.5 Thesis Outline

The rest of the thesis is organized into five chapters:

Chapter 2 provides a global overview on the topic of deep multi-task learning, by presenting major research trends. It then more specifically presents existing works in deep MTL optimization, which is the main topic of the thesis.

Chapter 3 introduces Maximum Roaming, a randomly dynamic partitioning method, aiming at creating more regularization while benefiting of the easier optimization provided by partitioning approaches. After a preliminary formulation of the partitioning strategy applied to the network, it presents the main contribution of the chapter, which is a novel update strategy for the partitioning scheme.

Chapter 4 focuses on the optimization strategy generally used by existing partitioning methods, through an extensive study and experimental evaluation, it shows how this strategy differs from most existing MTL approaches. The chapter then presents a novel optimization strategy, designed for state-of-the-art moment-based optimizers generally used with deep networks.

Chapter 5 demonstrates the relevance of the methods presented in Chapters 3 and 4 through their application in a real-world application. Concretely, it develops a MTL framework for glaucoma disease diagnosis from Retinal Fundus Images.

Finally, a conclusion of this thesis is given in Chapter 6. All the contributions brought in this work are summed up, and possible openings for further research are discussed from the current limitations in this work.



# Literature Review

---

## Contents

<b>2.1 Tackling task interference in Deep Multi-Task Learning . .</b>	<b>11</b>
2.1.1 Task affinities in MTL . . . . .	11
2.1.2 Network Architectures in MTL . . . . .	13
2.1.3 Multi-task Optimization . . . . .	14
<b>2.2 Related works in MTL optimization . . . . .</b>	<b>15</b>
2.2.1 Loss weighting . . . . .	15
2.2.2 Multi-Objective Optimization . . . . .	16
2.2.3 Gradient editing . . . . .	17
2.2.4 Parameter partitioning . . . . .	18

---

This chapter provides a literature review of deep Multi-Task Learning. In a first part, an overview of the main research trends in deep MTL is presented. In a second part, the focus is given to existing optimization strategies for deep MTL, which is the core of this thesis.

## 2.1 Tackling task interference in Deep Multi-Task Learning

Strategies to tackle task interference in MTL can be essentially classified into three main types of approaches: task affinities [Doersch & Zisserman 2017, Zamir *et al.* 2018, Strezoski *et al.* 2019b, Standley *et al.* 2020], networks architectures [Misra *et al.* 2016, Kokkinos 2017, Meyerson & Miikkulainen 2017, Rosenbaum *et al.* 2017] and multi-task optimization methods [Chen *et al.* 2018, Sener & Koltun 2018, Yu *et al.* 2020, Maninis *et al.* 2019, Bragman *et al.* 2019].

### 2.1.1 Task affinities in MTL

To maximize the benefits of multi-task learning and tackle task interference, some works propose to combine multiple tasks and compare the obtained performance to that one of the equivalent single task models, while keeping track of which combination is beneficial and which one produces degradation, i.e. task interference [Doersch & Zisserman 2017, Martinez Alonso & Plank 2017, Bingel & Søgaard 2017, Standley *et al.* 2020]. The objective is to discover some



multi-task affinities, that is which task combinations should be preferred to avoid task interference.

In [Doersch & Zisserman 2017] the multi-task networks systematically produce improvement compared to single-task ones, on a set of four computer vision tasks: position regression, colorization, motion segmentation, and exemplar matching. Instead, in [Martinez Alonso & Plank 2017] and [Bingel & Sogaard 2017], 1440 and 90 combinations of NLP tasks are tested respectively, with some task combinations clearly producing a performance degradation. In [Bingel & Sogaard 2017], a logistic regression model is then trained on these results to predict whether a task combination might or not be beneficial. The downside of these approaches is that training each combination to evaluate it has a prohibitive cost for deep networks. To avoid this computational burden, in [Strezoski *et al.* 2019b], the authors do not explicitly evaluate the task affinities. Instead, they progressively separate the tasks in different network branches during training to avoid task interferences, on the basis of a similarity computed between the gradients issued from the different tasks.

While multi-task affinities have not been largely explored in MTL, they have been thoroughly studied in Transfer Learning [Zamir *et al.* 2018, Dwivedi & Roig 2019, Song *et al.* 2019], with the similar objective of determining which task transfers can produce the most benefits. After introducing the Taskonomy dataset<sup>1</sup>, a CV dataset containing 4 million images from 600 different buildings with 26 annotated CV tasks, the authors in [Zamir *et al.* 2018] establish a transfer affinity graph by experimenting different transfers between the available tasks. As previously stated, this process is expensive, and requires close to 50000 GPU hours. [Dwivedi & Roig 2019] proposed a more computationally efficient method that evaluates the transfer affinity between two tasks by comparing their learned features when trained in isolation (single task). The assumption here is that tasks with high transfer affinity should learn similar features when trained in isolation. The comparison is conducted with Representation Similarity Analysis (RSA) [Kriegeskorte *et al.* 2008], a method coming from neuroscience, and used in deep networks to compare neural network activations. Finally, [Song *et al.* 2019] propose to compare attribution maps of a same input image with respect to the different tasks. The attribution maps associate to each pixel a relevance score with respect to the network output. The authors consider that tasks with high transfer affinities should essentially focus on the same parts of an image.

Despite the positive results reported in the literature, experimental results in [Standley *et al.* 2020] show that there is no evident correlation between transfer affinity and MTL affinity. This suggests that insights obtained from previous works on transfer affinities might not hold for MTL affinities, and that similar studies should be conducted for MTL. This work [Standley *et al.* 2020] also shows that the

---

<sup>1</sup><http://taskonomy.stanford.edu/>

performance gain with respect to single task approaches varies widely depending on the training settings. The latter suggests that task interference is not entirely determined by the nature of task pairings, and that advances in multi-task optimization might lead to systematic generalization benefits regardless of the task pairings.

### 2.1.2 Network Architectures in MTL

A large body of the literature in Multi-Task Learning focuses on designing better performing network architectures. Some works have proposed to enlarge deep networks with task-specific branches and attention mechanisms [Gao *et al.* 2019, Liu *et al.* 2019b, Misra *et al.* 2016, Mordan *et al.* 2018], giving tasks more room for specialization. In Cross-stitch networks [Misra *et al.* 2016], linear combinations are applied between parallel features of task-specific networks at different network depths. In a similar spirit, NDDR-CNN [Gao *et al.* 2019] replaces the linear combinations with  $1 \times 1$  convolutions. In [Mordan *et al.* 2018], Residual Auxiliary Blocks are incorporated in a network by directly using supervision from auxiliary tasks.

Other works adopt fine-grained architectural adaptations to fit a specific set of tasks [He *et al.* 2017, Xu *et al.* 2018, Zhang *et al.* 2019, Vandenhende *et al.* 2020]. Among these, there is Mask-RCNN [He *et al.* 2017] a notable pipeline for jointly processing instance detection, classification and segmentation. Arguing that task-specific prediction might be useful for performing other tasks, PAD-Net [Xu *et al.* 2018], Pattern Affinitive Propagation (PAP) [Zhang *et al.* 2019] and MTI-Net [Vandenhende *et al.* 2020] compute preliminary task predictions, which are then mixed together to produce final predictions. In [Zhang *et al.* 2019], an affinity learning layer is added for efficient combination of the task-specific predictions. Instead, in MTI-Net [Vandenhende *et al.* 2020], these affinities are computed at different feature scales. These works [Xu *et al.* 2018, Zhang *et al.* 2019, Vandenhende *et al.* 2020] have been designed for NYUv2 [Silberman *et al.* 2012], an indoor scene understanding dataset. However, such developments require important human efforts, domain-specific knowledge and development time.

A different family of works takes inspiration from Neural Architecture Search [Elsken *et al.* 2019] and Meta-Learning [Huisman *et al.* 2021], and propose dynamically evolving architectures, so that models can eventually modify their structure to avoid cases of task interference. In [Lu *et al.* 2017a], similarly to [Strezoski *et al.* 2019b], an initially fully shared network is progressively split layer by layer, starting from deepest layers, separating task groups based on the prediction errors faced over same data inputs. In [Meyerson & Miikkulainen 2017] and [Liang *et al.* 2018], each layer output for each task is computed as a learned task-specific weighting of different modules. [Gao *et al.* 2020] proposed a Gradient-based Neural Architecture Search (NAS) method, MTL-NAS, which learns to mix

different single-task architectures with feature fusions and then converges to a smaller fixed architecture. Authors in [Fernando *et al.* 2017] create a large shared network, from which subnetworks are optimized for each task through a genetic algorithm. More recently, in [Sun *et al.* 2019] the network learns for each task whether a layer shall be used or ignored, with the help of Gumbel-Softmax sampling [Maddison *et al.* 2017, Jang *et al.* 2017]. Inspired from [Bengio *et al.* 2013], in which a router adapts the network architecture with respect to each input sample, works like [Rosenbaum *et al.* 2017] and [Ahn *et al.* 2019] propose similar approaches for multi-task networks. The routers in these works are trained with Reinforcement Learning.

Similarly to works based on task affinities, NAS methods essentially aim at avoiding learning configurations which could lead to task interference. Although generally achieving great improvements with respect to task interference, many of these methods demand important resources, in terms of computational time or model size, due to factors like joint architecture and weights learning [Meyerson & Miikkulainen 2017, Liang *et al.* 2018], over-parameterization [Fernando *et al.* 2017], added task-specific branches [Misra *et al.* 2016, Liu *et al.* 2019b, Gao *et al.* 2020] or growing models [Lu *et al.* 2017b, Strezoski *et al.* 2019b]. As a consequence, they might not scale well with the number of tasks.

### 2.1.3 Multi-task Optimization

In opposition to the task affinities works which exploit task groupings to avoid optimization difficulties, and Architecture Search works which try to find the best network architecture for a given Multi-Task problem, a third stream of works aims at improving the optimization schemes for multi-task networks, in order to better exploit the inductive bias created by the plurality of tasks, regardless of the network architecture or task affinities. These works mainly involve loss weighting strategies [Chen *et al.* 2018, Guo *et al.* 2018, Kendall *et al.* 2018, Sinha *et al.* 2018, Liu *et al.* 2019a, Liu *et al.* 2019b], Multi-Objective Optimization [Sener & Koltun 2018, Lin *et al.* 2019], gradient editing [Chen *et al.* 2020, Yu *et al.* 2020] and parameter partitioning [Mallya *et al.* 2018, Mancini *et al.* 2018, Strezoski *et al.* 2019a, Maninis *et al.* 2019, Bragman *et al.* 2019]. The methods proposed by this type of techniques generally present multiple advantages:

- they are more computationally efficient, since they do not need to compute any prior knowledge on the tasks, nor do they modify the network architecture.
- they scale better to the number of tasks, for the same reasons.
- they generalize better to other problem settings (i.e. other domain, task, or network), since they are essentially domain, task and model-agnostic.

This thesis focuses on optimization schemes in deep Multi-Task Learning. In the next section a detailed review of this family of works is presented.

## 2.2 Related works in MTL optimization

This section presents the works most related to this thesis, which are optimization techniques for deep Multi-Task Learning. Methods under this category can be classified into three sub-categories: loss weighting, gradient editing and parameter partitioning strategies.

### 2.2.1 Loss weighting

As stated in Chapter 1, most MTL works optimize an aggregated objective function  $\mathcal{L}$  containing all the task-specific objective functions [Caruana 1997], generally under the form:

$$\mathcal{L} = \sum_i \omega_i \mathcal{L}_i \quad (2.1)$$

with  $\mathcal{L}_i$  the objective function of task  $i$ , and  $\omega_i$  its associated weight. These weights are of key importance to avoid high magnitude differences among the objective functions of the different tasks, which could harm the learning of low magnitude tasks. Tuning these weights with a grid-search presents two major issues:

- it is time consuming: the number of weights combinations to try exponentially grows with the number of tasks.
- changing dynamics during learning might require a dynamic weighting scheme.

Some deep MTL works have thus proposed different mechanisms to automatically and dynamically adapt these weights during the learning process [Chen *et al.* 2018, Guo *et al.* 2018, Kendall *et al.* 2018, Sinha *et al.* 2018, Liu *et al.* 2019a, Liu *et al.* 2019b].

In [Kendall *et al.* 2018] each task loss coefficient is expressed as a function of trainable task-specific prediction uncertainty parameters. While the model is pushed to reduce the overall uncertainty, the greater one task uncertainty is, the larger its associated weight. In [Liu *et al.* 2019a] these coefficients are modulated considering the learning speed (i.e. the rate of loss change) of each task. The idea behind it is that tasks whose losses decreased the least should get prioritized with larger weights. The coefficient  $\omega_i$  for task  $i$  at training step  $t$  is thus expressed as:

$$\omega_i(t) = \left( \frac{\mathcal{L}_i(t)}{\mathcal{L}_i(0)} \right)^\alpha \quad (2.2)$$

with  $\alpha$  a hyperparameter to tune.

Dynamic Weight Averaging (DWA) introduced in [Liu *et al.* 2019b] is a similar weighting scheme which considers loss ratios with respect to the previous learning step, i.e.  $\mathcal{L}_i(t-1)$  instead of  $\mathcal{L}_i(0)$  in Eq. 2.2. These ratios are used inside of a softmax with temperature scaling to progressively increase their contribution in the

final task weighting coefficients. [Guo *et al.* 2018] used a similar learning speed for the weighting, with the difference being that it is computed based on performance metrics instead of loss functions.

GradNorm [Chen *et al.* 2018] introduces a differentiable loss term aiming to match each task gradient magnitude with some desired magnitudes. These desired magnitudes are established based on the average task loss gradient and the learning speed of each task. Specifically, this loss term is defined as:

$$\mathcal{L}_{grad}(\omega_i(t)) = \sum_j \|G_j(t) - \bar{G}(t) \times (r_i(t))^\alpha\|_1 \quad (2.3)$$

with  $G_j(t)$  and  $\bar{G}(t)$  respectively the weighted gradient from task  $j$  and the average weighted gradient at step  $t$ , and  $r_i(t)$  the relative learning speed of task  $i$  at step  $t$ .

Finally, in [Sinha *et al.* 2018], gradient magnitude equalization among tasks is explicitly enforced with adversarial training. Specifically, an auxiliary network is trained to classify from which task incoming gradients are issued. An adversarial loss term based on this classification is then introduced in the main loss, to force the different tasks to produce indistinguishable gradients.

Although they adopt different weighting strategies, all these works have in common the optimization of an aggregated objective function, introduced in [Caruana 1997]. Some other works, presented below, instead question this aggregation, and tackle task interference by giving more consideration to the task-specific objectives.

## 2.2.2 Multi-Objective Optimization

Multi-Objective Optimization [Kaisa 1999] consists in optimizing simultaneously a set of objective functions  $(\mathcal{L}_1(t), \mathcal{L}_2(t), \dots, \mathcal{L}_N(t))$ , which holds strong analogies with MTL. A characteristic feature of Multi-Objective Optimization is that it does not consider the existence of a unique optimal solution. Instead, it defines a set of solutions named pareto frontier, from which no improvement is possible on one task without degrading another.

The first application of gradient descent for Multi-Objective Optimization, the Multiple Gradient Descent Algorithm (MGDA) has been proposed in [Désidéri 2012], and guarantees convergence in pareto stationary solutions, i.e. solutions on the pareto frontier. In [Sener & Koltun 2018], an adaptation of this algorithm is proposed, MGDA-UB, which scales to the high dimensionality of deep networks, by optimizing an upper-bound of the MGDA objective. MGDA-UB computes for each learning step scaling factors for each task gradient which guarantees no degradation on any task. This work has been extended by [Lin *et al.* 2019] to obtain a set of solutions with different trade-offs among tasks. Similarly to MGDA, these methods ensure, under

reasonable assumptions, to converge into a Pareto optimal solution, from which no improvement is possible for one task without deteriorating another task.

On a more general note, the motivation behind Multi-Objective Optimization is that averaging the different objectives into a single one leads to an information loss, and that task individuality should instead be maintained through the optimization. However, as pointed out in [Crawshaw 2020], the methods proposed in [Sener & Koltun 2018] and [Lin *et al.* 2019] come down to optimizing an aggregated objective function with dynamic weighting, similarly to loss weighting methods. Furthermore, these methods present a greedy behavior, trying to improve every task for each learning step. In the case of multiple non-convex objectives, this could lead to stagnation into task-specific local minima, performing poorly on the other tasks. More generally, converging on the pareto frontier is not a sufficient guarantee for good multi-task performance in non-convex settings.

### 2.2.3 Gradient editing

Similarly to Multi-Objective Optimization methods, a recent line of works, denoted in this thesis as gradient editing methods, also highlight the importance of considering individual task-specific gradient components to tackle task interference [Chen *et al.* 2020, Yu *et al.* 2020]. These works hypothesize that task interference is related to opposite gradient directions among the tasks. Therefore, they propose to modify the task-specific gradients to solve the conflicts before averaging them.

The method proposed in [Yu *et al.* 2020] takes inspiration in some Continual Learning works [Lopez-Paz & Ranzato 2017, Chaudhry *et al.* 2019]. The methods proposed in [Lopez-Paz & Ranzato 2017, Chaudhry *et al.* 2019], which learn sequentially incoming tasks, keep memories of previous task gradients for a given task  $j$ . For any new training step for a given task  $i$  they detect a gradient conflict if the gradient from task  $i$  points in an opposite direction of any other previous task gradient:

$$\exists j < i, \quad G_i(t)^T G_j(t) < 0 \quad (2.4)$$

with  $G_i(t)$  and  $G_j(t)$  are the gradient vectors of the current task  $i$  and of a previous task  $j$  memory at update step  $t$ . In case of such conflict, in [Lopez-Paz & Ranzato 2017] a quadratic optimization problem is solved to find the closest possible replacement candidate for  $G_i(t)$  which would not conflict with any  $G_j(t)$ ,  $\forall j < i$ . In [Chaudhry *et al.* 2019] a relaxed version is proposed by only solving the conflict with a unique average gradient of the previous tasks. This comes down to a simple projection on the normal plane in case of initial conflict, which greatly improves computation time. [Yu *et al.* 2020] propose to apply the same approach in a multi-task setting. For each training step, each pair of conflicting gradients gets projected onto the normal planes of each other, so that the optimization follows a consensual descent direction. Theoretically, this method

provides a better convergence in some conflicting regions with strong unbalance between the different task loss magnitudes.

In [Chen *et al.* 2020], the authors propose a probabilistic framework to solve gradient conflicts. They propose to have a network layer, transparent during the forward pass, through which each task gradient component is given a certain probability to be kept. This probability gets higher the more the gradient component's sign is in accordance with the sign of other tasks gradient components. Concretely, a gradient positive sign purity  $P$  is computed:

$$P = \frac{1}{2} \left( 1 + \frac{\sum_i \nabla \mathcal{L}_i}{\sum_i |\nabla \mathcal{L}_i|} \right) \quad (2.5)$$

This gradient positive sign purity represents the "average positivity" of each component of task gradients, ranging from 0, when all task-specific gradient components are negative, to 1 if they are all positive. For each model parameter component  $k$ , its associated gradient component  $g_{i,k}$  from task  $i$  has a probability to be kept of:

$$\begin{cases} P_k & \text{if } g_{i,k} > 0 \\ (1 - P_k) & \text{if } g_{i,k} < 0 \end{cases} \quad (2.6)$$

and is set to zero otherwise. All the task-specific component are then average into a multi-task loss for back-propagation.

Despite the promising results and convergence guarantees, these works apply corrections in some specific cases, i.e. gradients pointing in opposite directions, assuming that these are systematically detrimental to learning. Such assumption cannot fully hold in non-convex settings (or would require more prior knowledge on the loss landscapes), and some of these happenings might occasionally lead to the discovery of better regions of the loss landscape.

### 2.2.4 Parameter partitioning

One last family of works proposes a relaxed version of the hard parameter sharing strategy: while a unique set of parameter is defined for every task as in *hard* parameter sharing, for each task is defined a specific usage of this set of parameters. Concretely, this specification takes the form of attention masks applied at the neuron-level (i.e. convolutional filters), allowing each task to select, modulate or discard the output of each neuron. Binary masks can be used for binary neuron selection, and real-valued masks for modulation. In the following, this task-specific selection of parameters with selection masks is denoted as *parameter partitioning*. The idea behind works using parameter partitioning [Mallya *et al.* 2018, Mancini *et al.* 2018, Strezoski *et al.* 2019a, Maninis *et al.* 2019, Bragman *et al.* 2019] is to allow tasks specify their own usage of the shared parameters, while potentially reducing cases of task interference.



The first of such works, Piggyback [Mallya *et al.* 2018], is a method to adapt a pre-trained network to related tasks. The pre-trained weights are frozen, while task-specific binary masks (i.e. partitions) are learned to allow the tasks to specify their own usage of parameters. The advantage of this approach is that each task can learn from the pre-trained parameters without altering them, so that new tasks can be added with no degradation for the previous others. A similar approach is proposed in [Mancini *et al.* 2018], where affine transformations of the shared features are produced from learned real-valued partitions. However as pointed out in [Mallya *et al.* 2018], since the shared parameters are frozen in these two methods, the tasks cannot benefit from each other learning to strengthen their inductive bias.

Unlike [Mallya *et al.* 2018], in [Strezoski *et al.* 2019a] a parameter partitioning method allowing for a joint training of the shared parameters with respect to the multiple tasks is proposed. For this, binary selection masks are randomly initialized and fixed from start as the different task-specific partitions, while the shared parts are optimized by using this partitioning. The parameter partitions are initialized with one hyper-parameter, which determines the proportion of the total parameters that should be selected for every task. The experimental results suggest that this method greatly reduces task interference, by reducing the average number of tasks optimizing the same parameter.

In [Maninis *et al.* 2019], task-specific Squeeze and Excitation (SE) modules [Hu *et al.* 2018] are proposed to optimize real-valued parameter partitions, trained along with the network. Squeeze and Excitation modules are small modules made of  $1 \times 1$  fully connected layers, which learn for each layer an attention map at the filter level, which is then applied to the layer output. To further reduce task interference, the authors in [Maninis *et al.* 2019] also use Residual Adapters [Rebuffi *et al.* 2018], which are residual blocks aiming at refining task-specific features on top of the shared ones, and an adversarial training similar to [Sinha *et al.* 2018] to make the gradients from each task indistinguishable.

Finally, in [Bragman *et al.* 2019], task-specific binary partitions are set along with a shared one, and trained using a Gumbel-Softmax distribution [Maddison *et al.* 2017, Jang *et al.* 2017] to avoid the discontinuities created by binary assignments. The intuition here is to separate convolutional filters between one "generalist" group, used by every task, and multiple task-specific "specialist" groups used by isolated tasks to obtain more specific features.

Although these methods are essentially used as optimization methods for hard parameter sharing multi-task pipelines, they can also be considered themselves as a new kind of parameter sharing, which compromises the memory efficiency of hard parameter sharing, and the flexibility of soft parameter sharing. Back to optimization considerations, interestingly, the task-specific usage of parameters in all these methods imposes to alternately apply independent update steps with respect to the different



tasks objective functions, instead of aggregating them into a single objective, which is a different optimization scheme than in most other MTL works, applying update steps on an aggregated loss. This specific optimization scheme is discussed in [Maninis *et al.* 2019] as a way to exploit the benefits of partitioning, but is however never studied in isolation, theoretically or experimentally. These considerations make partitioning methods particularly interesting to study. This thesis therefore takes inspiration from partitioning methods to develop new optimization and sharing schemes for MTL pipelines.

# Strengthening the inductive bias with a dynamic parameter partitioning

---

## Contents

<b>3.1</b>	<b>Motivation</b>	<b>21</b>
<b>3.2</b>	<b>Preliminaries</b>	<b>23</b>
3.2.1	Parameter Partitioning	24
3.2.2	Parameter Partitioning Initialization	24
<b>3.3</b>	<b>Maximum Roaming Multi-Task Learning</b>	<b>25</b>
<b>3.4</b>	<b>Experimental Results</b>	<b>27</b>
3.4.1	Datasets	27
3.4.2	Baselines	28
3.4.3	Facial Attributes Detection	29
3.4.4	Scene Understanding	33
<b>3.5</b>	<b>Discussion</b>	<b>37</b>
<b>3.6</b>	<b>Appendix</b>	<b>38</b>

---

The content of this chapter is based on: "Maximum Roaming Multi-Task Learning" [Pascal *et al.* 2021a], which was published as a conference paper in AAAI 2021.

## 3.1 Motivation

In most existing works in MTL, parameters are shared among all tasks indifferently, except these used for task-specific predictions. However, this setting can lead to cases of task interferences, degrading the performance compared to single task models, and thus canceling one of the expected benefits of multi-tasking, which is better generalization on new data. In this section current limitations in related works are presented.

To address the problem of task interferences, several works have proposed to enlarge deep networks with task specific parameters [Gao *et al.* 2019, He *et al.* 2017, Kokkinos 2017, Liu *et al.* 2019b, Lu *et al.* 2017a, Misra *et al.* 2016, Mordan *et al.* 2018], giving tasks more room for specialization, and thus achieving

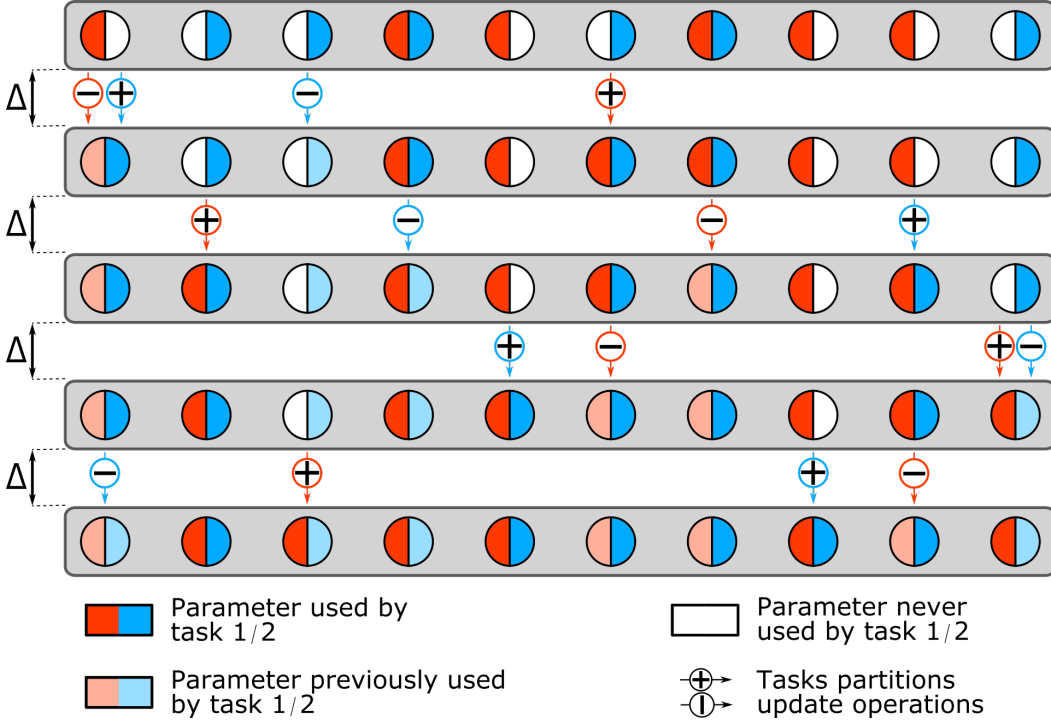


Figure 3.1: Maximum Roaming task partitions update process illustrated for two tasks in a layer containing 10 parameters. The partitions are initialized with a sharing ratio  $p = 0.6$ . After four update steps, every parameter has been used by both the tasks for at least  $\Delta$  iterations.

better results. Other works adopt architectural adaptations to fit a specific set of tasks [Xu *et al.* 2018, Zhang *et al.* 2018, Zhang *et al.* 2019, Vandenhende *et al.* 2020]. These approaches, however, do not solve the problem of task interference in the shared portions of the networks. Furthermore, they generally do not scale well with the number of tasks. Closer to the approach proposed in this chapter, some other works leave out architectural considerations and directly aim at easing the optimization in the shared parts of the networks, thanks to different loss weighting strategies [Kendall *et al.* 2018, Chen *et al.* 2018, Liu *et al.* 2019b, Sinha *et al.* 2018, Sener & Koltun 2018]. However most of these methods [Kendall *et al.* 2018, Chen *et al.* 2018, Liu *et al.* 2019b, Sinha *et al.* 2018] do not directly aim at addressing task interference, their main goal being to allow each task objective to have more or less magnitude in the main objective according to some learning dynamics. In the case of [Sener & Koltun 2018], although the method is guaranteed to converge in pareto-optimal solutions, the constraint of improving every single task at each training step is at risk in the case of non-convex loss landscapes, and can lead to quick convergence into poor loss regions in case of strongly interfering tasks. Finally, partitioning methods are explicitly used in [Strezoski *et al.* 2019a, Bragman *et al.* 2019, Maninis *et al.* 2019] to mitigate task-interferences by reducing the average number of tasks training a

given parameter: with this the chances are lower to having destructive conflicts on this parameter. However, despite the promising results, the risk in this strategy is to reduce the contribution of some tasks in the learning of each parameter, and therefore weaken the inductive bias of the model.

This chapter presents Maximum Roaming (Figure 3.1), a dynamic partitioning scheme that sequentially strengthens the inductive bias, while keeping task interference under control. Inspired by the dropout technique [Srivastava *et al.* 2014], the proposed method allows each parameter to *roam* across several task-specific sub-networks, thus giving them the ability to learn from a *maximum* number of tasks and build representations more robust to variations in the input domain. It can therefore be considered as a regularization method in the context of multi-task learning. Differently from other recent partitioning methods that aim at optimizing [Bragman *et al.* 2019, Maninis *et al.* 2019] or fixing [Strezoski *et al.* 2019a] a specific partitioning, maximum roaming privileges continuous random partition and assignment of parameters to tasks allowing them to learn from each task. Experimental results show consistent improvements over the state of the art methods.

The remaining of the chapter is organized as follows. Some preliminary elements and notations are first set out before the details of Maximum Roaming. Extensive experiments are then conducted to study the properties of the proposed method and to demonstrate its superior performance compared to other state-of-the-art MTL approaches. A final discussion over contributions and perspectives is presented.

## 3.2 Preliminaries

Let us define a training set  $\mathcal{T} = \{(\mathbf{x}_n, \mathbf{y}_{n,t})\}_{n \in [N], t \in [T]}$ , where  $T$  is the number of tasks and  $N$  the number of data points. The set  $\mathcal{T}$  is used to learn the  $T$  tasks with a standard shared convolutional network of depth  $D$  having one different final prediction layer for each task  $t$ . Under this setup, the convolutional filters of the network are referred to as *parameters*, with  $S^{(d)}$  the number of parameters of the  $d^{th}$  layer and  $i \in \{1, \dots, S^{(d)}\}$  its index. Finally,  $S_{max} = \max_d \{S^{(d)}\}$  represents the maximum number of parameters contained by a network layer.

In standard MTL, with fully shared parameters, the output of the  $d^{th}$  layer for task  $t$  is computed as:

$$f_t^{(d)}(H) = \sigma \left( H * K^{(d)} \right), \quad (3.1)$$

where  $\sigma(\cdot)$  is a non-linear function (e.g. ReLU),  $H$  a hidden input, and  $K^{(d)}$  the convolutional kernel composed of the  $S^{(d)}$  parameters of layer  $d$ .

### 3.2.1 Parameter Partitioning

Let us now introduce

$$\mathcal{M} = \left\{ \left( \mathbf{m}_1^{(d)}, \dots, \mathbf{m}_T^{(d)} \right) \right\}_{d \in [D]},$$

the parameter partitioning matrix, with  $\mathbf{m}_t^{(d)} \in \{0, 1\}^{S^{(d)}}$  a column vector associated to task  $t$  in the  $d^{th}$  layer, and  $m_{i,t}^{(d)}$  an element on such vector associated to the  $i^{th}$  parameter. As  $\mathcal{M}$  allows to select a subset of parameters for every  $t$ , the output of the  $d^{th}$  layer for task  $t$  (Eq. 3.1) is now computed as:

$$f_t^{(d)}(H_t) = \sigma \left( \left( H_t * K^{(d)} \right) \odot \mathbf{m}_t^{(d)} \right), \quad (3.2)$$

with  $\odot$  the channel-wise product. This notation is consistent with the formalization of the dropout (e.g. [Gomez *et al.* 2019]). By introducing  $\mathcal{M}$ , the hidden inputs are now also task-dependent: each task requires an independent forward pass, like in [Maninis *et al.* 2019, Strezoski *et al.* 2019a]. In other words, given a training point  $(\mathbf{x}_n, \{\mathbf{y}_{n,t}\}_{t=1}^T)$ , for each task  $t$  is computed an independent forward pass  $F_t(x) = f_t^{(D)} \circ \dots \circ f_t^{(1)}(\mathbf{x})$  and then back-propagate the associated task-specific losses  $\mathcal{L}_t(F_t(\mathbf{x}), \mathbf{y}_t)$ . Each parameter  $i$  receives independent training gradient signals from the tasks using it, i.e.  $m_{i,t}^{(d)} = 1$ . If the parameter is not used, i.e.  $m_{i,t}^{(d)} = 0$ , the received training gradient signals from those tasks account to zero.

For the sake of simplicity in the notation and without loss of generality, in the remaining of this document the index  $d$  indicating a given layer is omitted.

### 3.2.2 Parameter Partitioning Initialization

Every element of  $\mathcal{M}$  follows a Bernoulli distribution of parameter  $p$ :

$$P(m_{i,t} = 1) \sim \mathcal{B}(p).$$

Here  $p$  denotes the sharing ratio [Strezoski *et al.* 2019a]. The same value  $p$  is used for every layer of the network. The sharing ratio controls the overlap between task partitions, i.e. the number of different gradient signals a given parameter  $i$  will receive through training. Reducing the number of training gradient signals reduces task interference, by reducing the probability of having conflicting signals, and eases optimization. However, reducing the number of task gradient signals received by  $i$  also reduces the amount and the quality of inductive bias that different task gradient signals provide, which is one of the main motivations and benefits of multi-task learning [Caruana 1997].

A condition is imposed to guarantee the full capacity use of the network:

$$\sum_{t=1}^T m_{i,t} \geq 1. \quad (3.3)$$

Parameters not satisfying this constraint are attributed to a unique uniformly sampled task. The case  $p = 0$ , thus corresponds to a fully disjoint parameter partitioning, i.e.  $\sum_{t=1}^T m_{i,t} = 1, \forall i$ , whereas  $p = 1$  is a fully shared network, i.e.  $\sum_{t=1}^T m_{i,t} = T, \forall i$ , equivalent to Eq. 3.1.

Following a strategy similar to dropout [Srivastava *et al.* 2014], which forces parameters to successively learn efficient representations in many different randomly sampled sub-networks, the aim here is to make every parameter  $i$  learn from every possible task by regularly updating the parameter partitioning  $\mathcal{M}$ , i.e. make parameters *roam* among tasks to sequentially build the inductive bias, while still taking advantage of the "simpler" optimization setup regulated by  $p$ . For this Maximum Roaming Multi-Task Learning is introduced. It is a learning strategy consisting of two core elements: 1) a parameter partitioning update plan that establishes how to introduce changes in  $\mathcal{M}$ , and 2) a parameter selection process to identify the elements of  $\mathcal{M}$  to be modified.

### 3.3 Maximum Roaming Multi-Task Learning

In this section is formalized the core of the proposed contribution. Let first present an assumption that relaxes what can be considered as inductive bias.

**Assumption 1.** *The benefits of the inductive bias provided by the simultaneous optimization of parameters with respect to several tasks can be obtained by a sequential optimization with respect to different subgroups of these tasks.*

This assumption is in line with [Yosinski *et al.* 2014], where the authors state that initializing the parameters with transferred weights can improve generalization performance, and with other works showing the performance gain achieved by inductive transfer (see [He *et al.* 2017, Singh 1992, Tajbakhsh *et al.* 2016, Zamir *et al.* 2018]).

Assumption 1 allows to introduce the concept of evolution in time of the parameters partitioning  $\mathcal{M}$ , by indexing over time as  $\mathcal{M}(c)$ , where  $c \in \mathbb{N}$  indexes update time-steps, and  $\mathcal{M}(0)$  is the partitioning initialization. At every step  $c$ , the values of  $\mathcal{M}(c)$  are updated, under constraint (3.3), allowing parameters to roam across the different tasks.

**Definition 1.** *Let  $A_t(c) = \{i \mid m_{i,t}(c) = 1\}$  be the set of parameter indices used by task  $t$ , at update step  $c$ , and  $B_t(c) = \cup_{l=1}^c A_t(l)$  the set of parameter indices that have been visited by  $t$ , at least once, after  $c$  update steps. At step  $c + 1$ , the binary parameter partitioning matrix  $\mathcal{M}(c)$  is updated according to the following update rules:*

$$\begin{cases} \mathbf{m}_{i_-,t}(c+1) = 0, & i_- \in A_t(c) \\ \mathbf{m}_{i_+,t}(c+1) = 1, & i_+ \in \{1, \dots, S\} \setminus B_t(c) \\ \mathbf{m}_{i,t}(c+1) = \mathbf{m}_{i,t}(c), & \forall i \notin \{i_-, i_+\} \end{cases} \quad (3.4)$$

with  $i_+$  and  $i_-$  unique, uniformly sampled in their respective sets at each update step.

The frequency at which  $\mathcal{M}(c)$  is updated is governed by  $\Delta$ , where  $c = \lfloor \frac{E}{\Delta} \rfloor$  and  $E$  denotes the training epochs. This allows parameters to learn from a fixed partitioning over  $\Delta$  training iterations in a given partitioning configuration.  $\Delta$  has to be significantly large (it is expressed in terms of training epochs), so the network can fully adapt to each new configuration. Considering that discrete updates are applied in the parameter space, which has an impact in model performance, only one parameter is updated per update step to minimize the short-term impact. Figure 3.1 illustrates the full update process for one layer.

**Lemma 1.** *Any update plan as in Def.1, with update frequency  $\Delta$  has the following properties:*

1. *The update plan finishes in  $\Delta(1 - p)S_{max}$  training steps.*
2. *At completion, every parameter has been trained by each task for at least  $\Delta$  training epochs.*
3. *The number of parameters attributed to each task remains constant over the whole duration of update plan.*

**Proof:** Point 1 comes from the fact that  $B_t(c)$  grows by 1 at every step  $c$ , until all possible parameters in a given layer  $d$  are included, thus no new  $i_+$  can be sampled. At initialization,  $|B_t(c)| = pS$ , and it increases by one every  $\Delta$  training iterations, which gives the indicated result, upper bounded by the layer containing the most parameters. Point 2 is straightforward, since each new parameter partition remains frozen for at least  $\Delta$  training epochs. The same holds for item 3, since every update consists in the exchange of parameters  $i_-$  and  $i_+$   $\square$

Definition 1 requires to select update candidate parameters  $i_+$  and  $i_-$  from their respective subsets (Eq 3.4).  $i_+, i_-$  are both selected under a uniform distribution (without replacement), a lightweight solution to guarantee a constant overlap between the parameter partitions of the different tasks.

**Lemma 2.** *The overlap between parameter partitions of different tasks remains constant, on average, when the candidate parameters  $i_-$  and  $i_+$ , at every update step  $c + 1$ , are sampled without replacement under a uniform distribution from  $A_t(c)$  and  $\{1, \dots, S\} \setminus B_t(c)$ , respectively.*

**Proof:** It is proven by induction that  $P(m_{i,t}(c) = 1)$  is constant over  $c$ ,  $i$  and  $t$ , which ensures a constant overlap between the parameter partitions of the different tasks. The detailed proof is provided in appendix 3.6  $\square$

The probability of a parameter  $i$  to have been used by task  $t$ , after  $c$  update steps can be formulated as:

$$P(i \in B_t(c)) = p + (1 - p)r(c) \quad (3.5)$$

where

$$r(c) = \left( \frac{c}{(1-p)S} \right), \quad c \leq (1-p)S \quad (3.6)$$

is the *update ratio*, which indicates the completion rate of the update process within a layer. The condition  $c \leq (1-p)S$  refers to the fact that there cannot be more updates than the number of available parameters. It is also a necessary condition for  $P(i \in B_t(c)) \in [0, 1]$ . The increase of this probability represents the increase in the number of visited tasks for a given parameter, which is what creates inductive bias, following Assumption 1.

The benefits of Maximum Roaming are formalized in the following proposition:

**Proposition 1.** *Starting from a random binary parameter partitioning  $\mathcal{M}(0)$  controlled by the sharing ratio  $p$ , Maximum Roaming maximizes the inductive bias across tasks, while controlling task interference.*

**Proof:** Under Assumption 1, the inductive bias is correlated to the averaged number of tasks having optimized any given the parameter, which is expressed by Eq. 3.5.  $P(i \in B_t(c))$  is maximized with the increase of the number of updates  $c$ , to compensate the initial loss imposed by  $p \leq 1$ . The control over task interference cases is guaranteed by Lemma 2  $\square$

## 3.4 Experimental Results

This section first describes the datasets and the baselines used for comparison. The presented Maximum Roaming MTL method is first evaluated on several problems. First are studied its properties such as the effects the sharing ratio  $p$ , the impact of the interval between two updates  $\Delta$  and the completion rate of the update process  $r(c)$  and the importance of having a random selection process of parameters for update. Finally, a benchmark is presented, comparing MR with the different baseline methods. All code, data and experiments are available on GitHub <sup>1</sup>.

### 3.4.1 Datasets

Three publicly available datasets are used in the experiments:

**Celeb-A.** The official release is used, which consists of more than 200k celebrities images, annotated with 40 different facial attributes. To reduce the computational burden and allow for faster experimentation, it is casted into a multi-task problem by grouping the 40 attributes into eight groups of spatially or semantically related attributes (*e.g.* eyes attributes, hair attributes, accessories..) and creating one attribute prediction task for each group. The composition of these



Tasks	Classes
Global	Attractive, Blurry, Chubby, Double Chin, Heavy Makeup, Male, Oval Face, Pale Skin, Young
Eyes	Bags Under Eyes, Eyeglasses, Narrow Eyes, Arched Eyebrows, Bushy Eyebrows
Hair	Bald, Bangs, Black Hair, Blond Hair, Brown Hair, Gray Hair, Receding Hairline, Straight Hair, Wavy Hair
Mouth	Big Lips, Mouth Slightly Open, Smiling, Wearing Lipstick
Nose	Big Nose, Pointy Nose
Beard	5 o' Clock Shadow, Goatee, Mustache, No Beard, Sideburns
Cheeks	High Cheekbones, Rosy Cheeks
Wearings	Wearing Earrings, Wearing Hat, Wearing Necklace, Wearing Necktie

Table 3.1: Class composition of each the tasks for the Celeb-A dataset.

groups is provided in Table 3.1.

### Cityscapes

The Cityscapes dataset [Cordts *et al.* 2016] contains 5000 annotated street-view images with pixel-level annotations from a car point of view. The seven main semantic segmentation tasks are considered, along with a depth-estimation regression task, for a total of 8 tasks.

### NYUv2

The NYUv2 dataset [Silberman *et al.* 2012] is a challenging dataset containing 1449 indoor images recorded over 464 different scenes from Microsoft Kinect camera. It provides 13 semantic segmentation tasks, depth estimation and surfaces normals estimation tasks, for a total of 15 tasks. As with Cityscapes, the pre-processed data provided by [Liu *et al.* 2019b] is used.

### 3.4.2 Baselines

Maximum Roaming is compared with several alternatives, including two parameter partitioning approaches [Maninis *et al.* 2019, Strezoski *et al.* 2019a]. Among these, [Bragman *et al.* 2019] was not included, as it was not possible to correctly replicate the method with the available resources. Specifically, the following methods are evaluated:

- MTL, a standard fully shared network with uniform task weighting.

<sup>1</sup><https://github.com/lucas-pascal/Maximum-Roaming-Multi-Task-Learning>

- GradNorm [Chen *et al.* 2018], a fully shared network with trainable task weighting method.
- MGDA-UB [Sener & Koltun 2018], a fully shared network which formulates the MTL as a multi-objective optimization problem.
- Task Routing (TR) [Strezoski *et al.* 2019a], a parameter partitioning method with fixed binary masks.
- SE-MTL [Maninis *et al.* 2019] a parameters partitioning method, with trainable real-valued masks. Note that it consists in the original paper of a more complex framework which comprises several other contributions. For a fair comparison with the other baselines, only the parameter partitioning is considered and not the other elements of their work.
- STL, the single-task learning baselines, using one model per task.

### 3.4.3 Facial Attributes Detection

These first experiments study in detail the properties of the proposed method using the Celeb-A dataset. Being a small dataset it allows for fast experimentation.

#### Experimental setup

Table 3.1 provides details on the distribution of the 40 facial attributes between the 8 created tasks. Every attribute in a task uses the same parameter partition. During training, the losses of all the attributes of the same task are averaged to form a task-specific loss. All baselines use a ResNet-18 [He *et al.* 2016c] truncated after the last average pooling as a shared network. 8 fully connected layers of input size 512 are then added, one per task, with the appropriate number of outputs, i.e. the number of facial attributes in the task. The partitioning methods ([Maninis *et al.* 2019], [Strezoski *et al.* 2019a] and Maximum Roaming) are applied to every shared convolutional layer in the network. The parameter  $\alpha$  in GradNorm [Chen *et al.* 2018] has been optimized in the set of values  $\{0.5, 1, 1.5\}$ . All models were trained with an Adam optimizer [Kingma & Ba 2017] and a learning rate of  $1e-4$ , until convergence, using a binary cross-entropy loss function, averaged over the different attributes of a given task. The batch size is set to 256, and all input images are resized to  $(64 \times 64 \times 3)$ . The reported results are evaluated using a validation split provided in the official release of the dataset [Liu *et al.* 2015b]. The reported results are averaged over five seeds.

#### Effect of Roaming

In a first experiment, the effects of the roaming imposed to parameters in MTL performance as a function of the sharing ratio  $p$  are studied, and compared with a fixed partitioning setup. Figure 3.2 reports achieved F-scores as  $p$  varies, with  $\Delta = 0.1$  and  $r(c) = 100\%$ . Let us remark that as all models scores are averaged over 5 seeds, this means that the fixed partitioning scores are the average of 5 different

(fixed) partitionings.

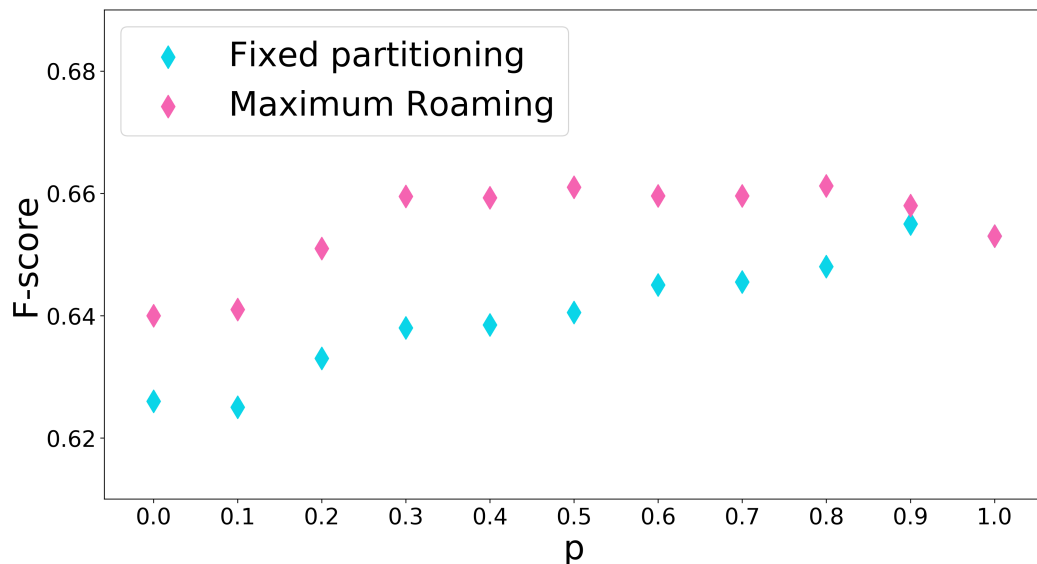


Figure 3.2: Contribution of Maximum Roaming depending on the parameter partitioning selectivity  $p$ .

Results show that for the same network capacity Maximum Roaming provides improved performance w.r.t. a fixed partitioning approach. Moreover, as the values of  $p$  are smaller, and for the same network capacity, Maximum Roaming does not suffer from a dramatic drop in performance as it occurs using a fixed partitioning. This behavior suggests that parameter partitioning does have an unwanted effect on the inductive bias that is, thus, reflected in poorer generalization performance. However, these negative effects can be compensated by parameter roaming across tasks.

The fixed partitioning scheme (blue bars) achieves its best performance at  $p = 0.9$  (F-score= 0.6552). This is explained by the fact that the dataset is not originally made for multi-task learning: all its classes are closely related, so they naturally have a lot to share with few task interference. Maximum Roaming achieves higher performance than this nearly full shared configuration (the overlap between task partitions is close to its maximum) for every  $p$  in the range  $[0.3, 0.9]$ . In this range, the smaller  $p$  is, the greater the gain in performance: it can be profitable to partially separate tasks even when they are very similar (i.e. multi-class, multi-attribute datasets) while allowing parameters to roam.

#### Effect of $\Delta$ and $r(c)$

Here, the impact of the interval between two updates  $\Delta$  and the completion rate of the update process  $r(c)$  (Eq. 3.6) is studied. Using a fixed sharing ratio,  $p = 0.5$ ,

the obtained F-score values of Maximum Roaming over a grid search with respect to these two hyper-parameters are reported in Figure 3.3(center).

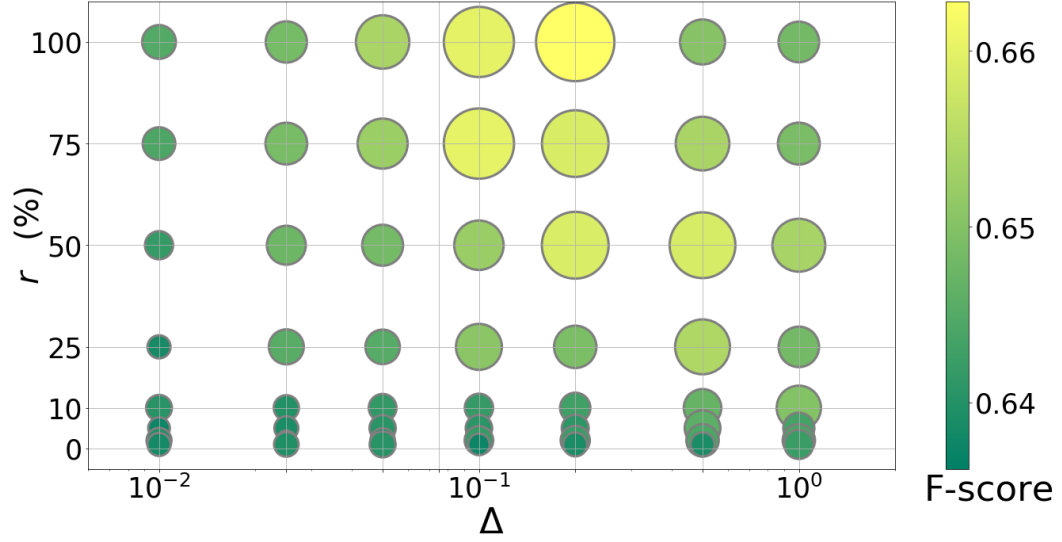


Figure 3.3: F-score of the proposed method reported for different values of the update interval  $\Delta$  and the update completion rate  $r$ . Different colors and circle sizes indicate different F-score values.

Results show that the model’s performance increases for a wide range of  $\Delta$  values ( $\sim 0.05$ -1 epochs). For higher  $\Delta$  values, the update process is still going on while the model starts to overfit, which seems to prevent it from reaching its full potential. A rough knowledge of the overall learning behavior on the training dataset or a coarse grid search is enough to set it. Regarding the completion percentage  $r$ , as it would be expected, the F-score increases with  $r$  as long as  $\Delta$  is not too high. The performance improvement becomes substantial beyond  $r = 25\%$ , suggesting that it can also be tuned to adapt the duration of the update process without incurring in a significant loss.

#### Role of random selection

Finally, the importance of choosing candidate parameters for updates under a uniform distribution is addressed. To this end, here is defined a deterministic selection process to systematically choose  $i_-$  and  $i_+$  within the update plan of Def. 1. New candidate parameters are selected to minimize the average cosine similarity in the task parameter partition. The intuition behind this update plan is to select parameters which are the most likely to provide additional information for a task, while discarding the more redundant ones based on their weights. The candidate

parameters  $i_-$  and  $i_+$  are thus respectively selected such that:

$$i_- = \arg \min_{u \in A_t(c)} \left( \sum_{v \in (A_t(c) \setminus \{u\})} \frac{K_u \cdot K_v}{\|K_u\| \|K_v\|} \right)$$

$$i_+ = \arg \max_{u \in \{1, \dots, S\} \setminus B_t(c)} \left( \sum_{v \in A_t(c)} \frac{K_u \cdot K_v}{\|K_u\| \|K_v\|} \right)$$

with  $K_u, K_v$  the parameters  $u, v$  of the convolutional kernel  $K$ . Figure 3.4 compares this deterministic selection process with Maximum Roaming by reporting the best F-scores achieved by the fully converged models for different completion rates  $r(c)$  of the update process.

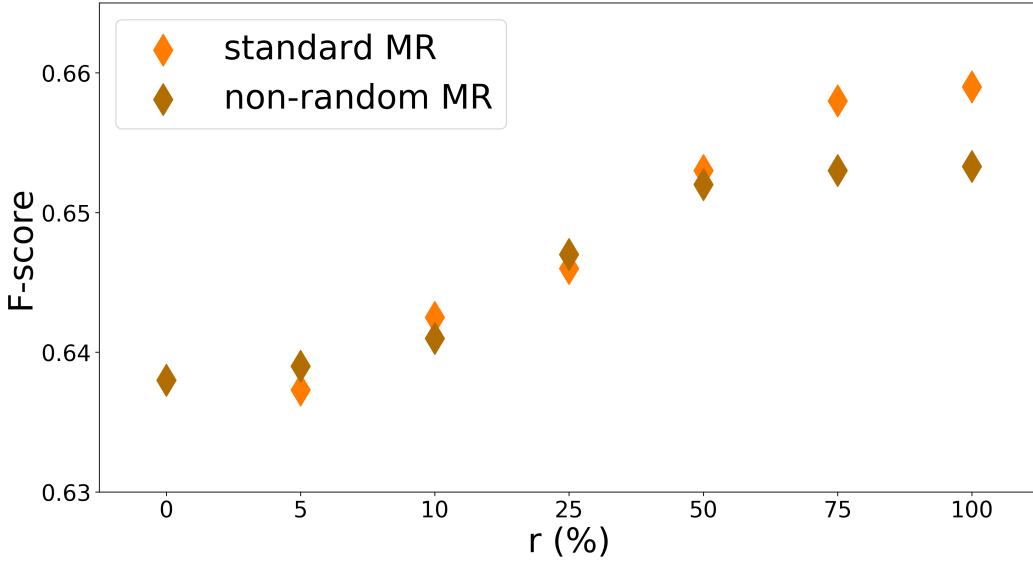


Figure 3.4: Comparison of Maximum Roaming with random and non-random selection process of parameter candidates for updates.

Results show that, while both selection methods perform about equally at low values of  $r$ , MR progressively improves as  $r$  grows. This is to attribute to the varying overlapping induced by the deterministic selection. Thanks to it, outliers in the parameter space have more chances than others to be quickly selected as update candidates, which slightly favours a specific update order, common to every task. This has the effect of increasing the overlap between the different task partitions, along with the cases of task interference.

It should be noted that the deterministic selection method still provides a significant improvement compared to a fixed partitioning ( $r = 0$ ). This highlights the primary importance of making the parameters learn from a maximum number of tasks, which is guaranteed by the update plan (Def. 1), i.e. the *roaming*, used by both selection methods.

### Benchmark

Finally, the proposed method is benchmarked with the different baselines. Precision, recall and f-score metrics averaged over the 40 facial attributes are reported, along with the average ranking of each MTL model over the reported performance measures; and the ratio  $\#P$  of trainable parameters w.r.t. the MTL baseline (Table 3.2). The partitioning methods (TR, SE-MTL and MR) achieve the three best results, and the MR method performs substantially better than the two others.

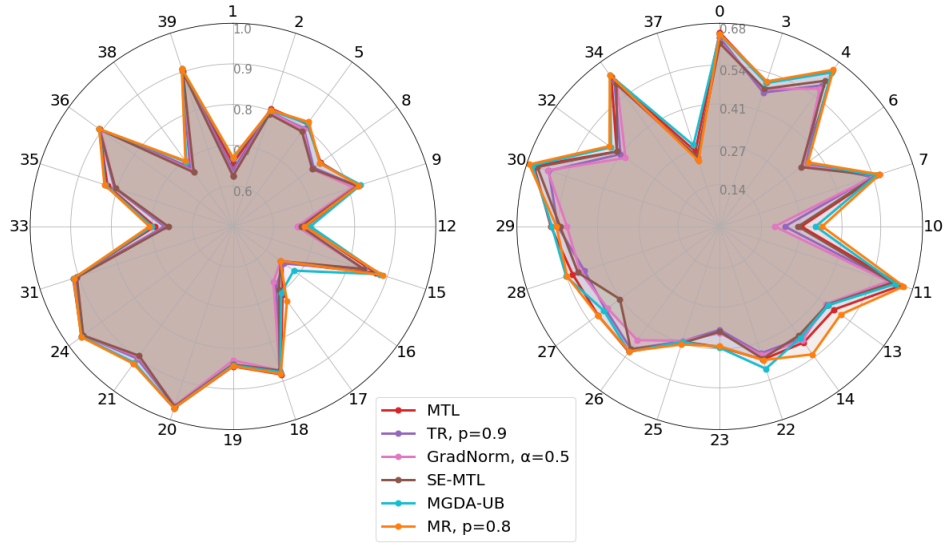


Figure 3.5: Radar chart comparing different baselines F-scores on every facial attribute of Celeb-A. (left) attributes with highest scores, (right) attributes with lowest scores. Each plot is displayed at a different scale.

On top of the benchmark, Figure 3.5 shows radar charts with the individual F-scores obtained by the different multi-task baselines for each of the 40 facial attributes. For improved readability, the scores have been plotted in two different charts, one for the 20 highest scores and one for the remaining 20 lowest. Results confirm the superiority of Maximum Roaming (already shown in Table 3.2), and show the consistency of these observations across the 40 classes, the MR model reaching the best performances on several individual facial attributes.

#### 3.4.4 Scene Understanding

This experiment compares the performance of MR with the baseline methods in two well-established scene-understanding benchmarks: Cityscapes and NYUv2.

#### Experimental setup

For this study, each segmentation task is considered as an independent task, although it is a common approach to consider all of them as a unique task. As with the

	#P	Multi-Attribute Classification			
		Precision ( $\uparrow$ )	Recall ( $\uparrow$ )	F-Score ( $\uparrow$ )	Rank ( $\downarrow$ )
STL	7.9	67.10 ( $\pm 0.37$ )	61.99 ( $\pm 0.49$ )	64.07 ( $\pm 0.21$ )	-
MTL	1.0	68.67 ( $\pm 0.69$ )	59.54 ( $\pm 0.52$ )	62.95 ( $\pm 0.21$ )	5.33
GradNorm ( $\alpha = 0.5$ )	1.0	70.36 ( $\pm 0.07$ )	59.49 ( $\pm 0.58$ )	63.55 ( $\pm 0.49$ )	5.00
MGDA-UB	1.0	68.64 ( $\pm 0.12$ )	60.21 ( $\pm 0.33$ )	63.56 ( $\pm 0.27$ )	4.66
SE-MTL	1.1	71.10 ( $\pm 0.28$ )	62.64 ( $\pm 0.51$ )	65.85 ( $\pm 0.17$ )	2.33
TR ( $p = 0.9$ )	1.0	<b>71.71</b> ( $\pm 0.06$ )	61.75 ( $\pm 0.47$ )	65.51 ( $\pm 0.32$ )	2.33
MR ( $p = 0.8$ )	1.0	71.24 ( $\pm 0.35$ )	<b>63.04</b> ( $\pm 0.56$ )	<b>66.23</b> ( $\pm 0.20$ )	<b>1.33</b>

Table 3.2: Celeb-A results (Average over 40 facial attributes). The best per column score of an MTL method is underlined.

Celeb-A dataset, for the sake of fairness in comparison, all approaches use the same base network, a SegNet [Badrinarayanan *et al.* 2017] outputting 64 feature maps of same height and width as the inputs. For each of the 8 tasks, one prediction head is added, composed of one ( $3 \times 3 \times 64 \times 64$ ) and one ( $1 \times 1 \times 64 \times 1$ ) convolutions. A sigmoid function is applied on the output of the segmentation tasks. The partitioning methods ([Maninis *et al.* 2019], [Strezoski *et al.* 2019a] and Maximum Roaming) are applied to every shared convolutional layer in the network. This excludes those in the task respective prediction heads. The parameter  $\alpha$  in GradNorm [Chen *et al.* 2018] has been optimized in the set of values  $\{0.5, 1, 1.5\}$ . All models were trained with an Adam optimizer [Kingma & Ba 2017] and a learning rate of  $1e-4$ , until convergence. The binary cross-entropy is used as a loss function for each segmentation task, and the averaged absolute error for the depth estimation task. For the normals estimation task of NYUv2, the prediction head is made of one ( $3 \times 3 \times 64 \times 64$ ) and one ( $1 \times 1 \times 64 \times 3$ ) convolutions. Its loss is computed with an element-wise dot product between the normalized predictions and the ground-truth map. The batch size is set to 8 for Cityscapes and 2 for NYUv2. The input samples are resized to  $128 \times 256$  for Cityscapes and  $288 \times 384$  for NYUv2, provided as such by [Liu *et al.* 2019b]<sup>2</sup>. The reported results are evaluated on the validation splits furnished by [Liu *et al.* 2019b].

<sup>2</sup><https://github.com/lorenmt/mtan>

**Benchmark**

The reported metrics are Intersection over Union (mIoU) and pixel accuracy (Pix. Acc.) averaged over all segmentation tasks, average absolute (Abs. Err.) and relative error (Rel. Err.) for depth estimation tasks, mean (Mean Err.) and median errors (Med. Err.) for the normals estimation task, the ratio #P of trainable parameters w.r.t. MTL, and the average rank of the MTL methods over the measures. STL is not included in the ranking, as it is considered of a different nature, but reported as a baseline reference.

Tables 3.3 and 3.4 report the results on Cityscapes and NYUv2, respectively. The reported results are the best achieved with each method on the validation set, averaged over 3 seeds, after a grid-search on the hyper-parameters.

	#P	Segmentation ( $\uparrow$ )		Depth estimation ( $\downarrow$ )		
		mIoU	Pix. Acc.	Abs. Err.	Rel. Err.	Rank ( $\downarrow$ )
STL	7.9	58.57 ( $\pm 0.49$ )	97.46 ( $\pm 0.03$ )	0.0141 ( $\pm 0.0002$ )	22.59 ( $\pm 1.15$ )	-
MTL	1.0	56.57 ( $\pm 0.22$ )	97.36 ( $\pm 0.02$ )	0.0170 ( $\pm 0.0006$ )	43.99 ( $\pm 5.53$ )	3.75
GradNorm ( $\alpha = 1.5$ )	1.0	56.77 ( $\pm 0.08$ )	<b>97.37</b> ( $\pm 0.02$ )	0.0199 ( $\pm 0.0004$ )	68.13 ( $\pm 4.48$ )	3.87
MGDA-UB	1.0	56.19 ( $\pm 0.24$ )	97.33 ( $\pm 0.01$ )	<b>0.0130</b> ( $\pm 0.0001$ )	<b>25.09</b> ( $\pm 0.28$ )	2.50
SE-MTL	1.1	55.45 ( $\pm 1.03$ )	97.24 ( $\pm 0.10$ )	0.0160 ( $\pm 0.0006$ )	35.72 ( $\pm 1.62$ )	4.87
TR ( $p = 0.6$ )	1.0	56.52 ( $\pm 0.41$ )	97.24 ( $\pm 0.04$ )	0.0155 ( $\pm 0.0003$ )	31.47 ( $\pm 0.55$ )	3.87
MR ( $p = 0.6$ )	1.0	<b>57.93</b> ( $\pm 0.20$ )	<b>97.37</b> ( $\pm 0.02$ )	0.0143 ( $\pm 0.0001$ )	29.38 ( $\pm 1.66$ )	<b>1.62</b>

Table 3.3: Cityscape results. The best per column score of an MTL method is underlined.

Maximum Roaming reaches the best scores on segmentation and normals estimation tasks, and ranks second on depth estimation tasks. In particular, it outperforms other methods on the segmentation tasks: it restores the inductive bias decreased by parameter partitioning, so the tasks benefiting the most from it are the ones most similar to each other, which are here the segmentation tasks. Furthermore, MR uses the same number of trainable weights than the MTL baseline, plus a few binary



	#P	Segmentation		Depth estimation		Normals estimation		Rank (↓)
		mIoU (↑)	Pix. Acc. (↑)	Abs. Err. (↑)	Rel. Err. (↑)	Mean Err. (↑)	Med. Err. (↑)	
STL	14.9	13.12 (±1.06)	94.58 (±0.14)	67.46 (±2.64)	28.79 (±1.18)	29.77 (±0.22)	23.93 (±0.15)	-
MTL	1.0	15.98 (±0.56)	94.22 (±0.25)	60.95 (±0.41)	<b>25.54</b> (±0.07)	32.43 (±0.19)	27.43 (±0.35)	3.7
GradNorm	1.0	16.13 (±0.23)	94.43 (±0.07)	76.26 (±0.34)	32.08 (±0.50)	34.45 (±0.52)	30.98 (±0.80)	4.5
MGDA-UB	1.0	2.96 (±0.35)	82.87 (±0.23)	186.9 (±15.3)	98.74 (±5.34)	46.96 (±0.37)	45.15 (±0.70)	6.0
SE-MTL	1.2	16.02 (±0.12)	94.56 (±0.01)	<b>59.88</b> (±1.12)	26.30 (±0.58)	32.22 (±0.02)	26.12 (±0.02)	2.7
TR ( $p = 0.8$ )	1.0	16.54 (±0.02)	94.58 (±0.11)	63.54 (±0.85)	27.86 (±0.90)	30.93 (±0.19)	25.51 (±0.28)	2.7
MR ( $p = 0.8$ )	1.0	<b>17.40</b> (±0.31)	<b>94.86</b> (±0.06)	60.82 (±0.23)	27.50 (±0.15)	<b>30.58</b> (±0.04)	<b>24.67</b> (±0.08)	<b>1.5</b>

Table 3.4: NYUv2 results. The best per column score of an MTL method is underlined.

partitions masks (negligible), which means it scales almost optimally to the number of tasks. This is also the case for the other presented baselines, which sets them apart from heavier models in the literature, which add task-specific branches in their networks to improve performance at the cost of scalability.

For other MTL baselines, it is observed that GradNorm fails on the regression tasks (depth and normals estimation). This is due to the equalization of the task respective gradient magnitudes. Specifically, since the multi-class segmentation task is divided into independent segmentation tasks (7 for Cityscapes and 13 for NYUv2), GradNorm attributes to the depth estimation task of Cityscapes only one eighth of the total gradient magnitude, which gives it a systematically low importance compared to the segmentation tasks which are more likely to agree on a common gradient direction, thus diminishing the depth estimation task. Instead, in MTL the gradient’s magnitude is not constrained, having more or less importance depending on the loss obtained for a given task. This explains why the regression tasks are better handled by this simpler model in this configuration. For instance, in a configuration with the CityScapes segmentation classes addressed as one task (for 2 tasks in total), GradNorm keeps its good segmentation performance and improves at regression tasks (see Table 3.3), which confirms the hypothesis. It is also observed that MGDA-UB reaches pretty low performance on the NYUv2

dataset, especially on segmentation tasks, while being one of the best performing ones on Cityscapes. It appears that during training, the loss computed for the shared weights quickly converges to zero, leaving task-specific prediction layers to learn their task independently from an almost frozen shared representation. This could also explain why it still achieves good results at the regression tasks, these being easier tasks. A hypothesis is that the solver fails at finding good directions improving all tasks, leaving the model stuck in a Pareto-stationary point.

When comparing to the single task learners counterpart, it is observed that on Cityscapes STL achieves slightly better segmentation performances than the other approaches, and competitive results on depth estimation. On NYUv2 (and Celeb-A), its results are far from the best MTL models. These show that complex setups proposing numerous tasks, as in the setup proposed here (8, 8 and 15), are challenging for the different MTL baselines, resulting in losses in performance as the number of tasks increase. This is not a problem with STL, which uses an independent model for each task. However, the associated increase in training time and parameters (15× more parameters for NYUv2, which is equivalent to 375M parameters) makes it inefficient in practice, while its results are not even guaranteed to be better than the multi-task approaches.

### 3.5 Discussion

This chapter introduced Maximum Roaming, a dynamic parameter partitioning method designed to reduce the task interference phenomenon while better exploiting the latent inductive bias represented by the plurality of tasks. The proposed approach makes each parameter learn successively from all possible tasks, with a simple yet effective parameter selection process. The proposed algorithm achieves it in a minimal time, without additional costs compared to other partitioning methods, nor additional parameter to be trained on top of the base network. Experimental results show a substantially improved performance on all reported datasets, regardless of the type of convolutional network it applies on.

From these encouraging results, multiple directions of improvement could be explored. First, from its formulation, one can notice that the update process finishes at different times depending on the layer depth, which may eventually weaken some co-adaptations when partitions get frozen. An update pace adapted to the layer depth may instead be more adapted. Similarly, an adaptation of the sharing ratio to the network depth could be considered, since convolutional networks representations get more specific as layers get deeper [Yosinski *et al.* 2014], suggesting that less conflicts should arise in the shallow parts of the network.

Finally, one shall notice that, similarly to other partitioning methods, Maximum Roaming proceeds task specific learning steps, which is slightly different from opti-

mizing a unique objective function aggregating the task specific ones, as most other MTL methods. This specificity is studied extensively in Chapter 4.

### 3.6 Appendix

#### Proof of Lemma 2

At  $c = 0$ , every element of  $\mathcal{M}(0)$  follows a Bernoulli distribution:

$$P(m_{i,t} = 1) \sim \mathcal{B}(p).$$

Lets assume  $P(m_{i,t}(c) = 1) = p, \quad \forall c \in \{1, \dots, (1-p)S - 1\}$  and prove it holds for  $c + 1$ .

The probability  $P(m_{i,t}(c + 1) = 1)$  can be written as:

$$\begin{aligned} P(m_{i,t}(c + 1) = 1) = \\ P(m_{i,t}(c + 1) = 1 \mid m_{i,t}(c) = 1)P(m_{i,t}(c) = 1) \\ + P(m_{i,t}(c + 1) = 1 \mid m_{i,t}(c) = 0)P(m_{i,t}(c) = 0). \end{aligned} \quad (3.7)$$

Since  $P(m_{i,t}(c) = 1) = P(i \in A_t(c))$ , Eq. 3.7 can be reformulated as:

$$\begin{aligned} P(i \in A_t(c + 1)) = \\ P(i \in A_t(c + 1) \mid i \in A_t(c))P(i \in A_t(c)) \\ + P(i \in A_t(c + 1) \mid i \notin A_t(c))P(i \notin A_t(c)). \end{aligned} \quad (3.8)$$

As  $i_-$  is uniformly sampled from  $A_t(c)$ , the first term in Eq. 3.8 can be reformulated as

$$\begin{aligned} P(i \in A_t(c + 1) \mid i \in A_t(c))P(i \in A_t(c)) = \\ \left(1 - \frac{1}{pS}\right)p = p - \frac{1}{S}. \end{aligned} \quad (3.9)$$

Let us now expand the second term in Eq. 3.8 by considering whether  $i \in B_t(c)$  or not:

$$\begin{aligned} P(i \in A_t(c + 1) \mid i \notin A_t(c))P(i \notin A_t(c)) = \\ P(i \in A_t(c + 1) \mid i \notin A_t(c), i \notin B_t(c)) \\ \times P(i \notin A_t(c) \mid i \notin B_t(c))P(i \notin B_t(c)) \\ + P(i \in A_t(c + 1) \mid i \notin A_t(c), i \in B_t(c)) \\ \times P(i \notin A_t(c) \mid i \in B_t(c))P(i \in B_t(c)). \end{aligned} \quad (3.10)$$

From Def. 1,  $P(i \in A_t(c + 1) \mid i \notin A_t(c), i \in B_t(c)) = 0$  and  $A_t(c) \subset B_t(c)$ , thus (3.10) becomes:

$$\begin{aligned} P(i \in A_t(c + 1) \mid i \notin A_t(c))P(i \notin A_t(c)) = \\ P(i \in A_t(c + 1) \mid i \notin B_t(c))P(i \notin B_t(c)). \end{aligned}$$

Given that  $i_+$  is uniformly sampled from  $\{1, \dots, S\} \setminus B_t(c)$  :

$$P(i \in A_t(c+1) \mid i \notin A_t(c))P(i \notin A_t(c)) = \frac{1}{(1-p)S-c} \cdot \frac{(1-p)S-c}{S} = \frac{1}{S}. \quad (3.11)$$

Then, by replacing (3.9) and (3.11) in Eq. 3.8:

$$\begin{aligned} P(m_{i,t}(c+1) = 1) &= P(i \in A_t(c+1)) \\ &= p - \frac{1}{S} + \frac{1}{S} \\ &= p, \end{aligned}$$

which demonstrates that  $P(m_{i,t}(c) = 1)$  remains constant over  $c$ , given a uniform sampling of  $i_-$  and  $i_+$  from  $A_t(c)$  and  $\{1, \dots, S\} \setminus B_t(c)$ , respectively  $\square$



# Separating task-specific objectives for a better optimization

---

## Contents

---

<b>4.1</b>	<b>Motivation</b>	<b>41</b>
<b>4.2</b>	<b>Alternate and independent optimization of task-specific objective functions</b>	<b>43</b>
4.2.1	Standard MTL optimization with aggregated loss	43
4.2.2	Alternate and independent optimization of task-specific objective functions for SGD	44
4.2.3	Alternate and independent optimization of task-specific objective functions for moving-average based optimizers	45
4.2.4	Mitigating computational costs through task grouping	46
<b>4.3</b>	<b>Experiments and results</b>	<b>47</b>
4.3.1	Scene understanding on NYUv2	47
4.3.2	Multi-class segmentation on Cityscapes	49
4.3.3	Multi-attribute segmentation on Celeb-A	50
4.3.4	Covered distance	52
<b>4.4</b>	<b>Discussion</b>	<b>53</b>
<b>4.5</b>	<b>Appendix</b>	<b>54</b>

---

The content of this chapter is based on "Optimization Strategies in Multi-Task Learning: Averaged or Independent Losses?" [Pascal *et al.* 2021b] (to be submitted).

## 4.1 Motivation

The optimization objective in MTL has been introduced for the first time in [Caruana 1997] as the combination into a single aggregated objective function, generally in the form of a weighted sum of all the task-specific objective functions, which the shared model is trained to minimize. This formulation is particularly convenient for deep MTL. First, it is computationally efficient, since a single gradient descent step (i.e. one forward and one backward propagation) optimizes the shared parameters w.r.t. every task, with the only extra computational cost coming from the task-specific parts of the network. Second, in the case of convex task-specific objective functions, it guarantees convergence to the optimum, since the average

## Chapter 4. Separating task-specific objectives for a better optimization

---

objective function remains convex.

Following works then developed new methods to mitigate the problems associated to the complexity of the loss [Vandenhende *et al.* 2021, Crawshaw 2020]. However, as detailed in Chapter 2, most of them rely on similar aggregated loss optimization schemes: a first family of methods uses dynamic loss weighting strategies to control the influence of tasks in the main objective function and to account for the learning dynamics of the different tasks [Chen *et al.* 2018, Guo *et al.* 2018, Kendall *et al.* 2018, Liu *et al.* 2019b]. Other works formulate the problem as a multi-objective optimization task [Désidéri 2012], which converges to a Pareto optimal solution, from which no task can be improved without hurting another one [Lin *et al.* 2019, Sener & Koltun 2018]. A more recent line of works proposes to modify the task-specific gradients before averaging them, when there are conflicting gradient directions, under the hypothesis that they can be destructive for some tasks [Chen *et al.* 2020, Yu *et al.* 2020]. Although some of these works provide a better convergence in specific conflicting regions of the loss landscape, the guarantees are very local and become of relative importance in the case of complex non-convex loss landscapes, for which a wider exploration of the parameter space might be preferred over greedy strategies.

On the other hand, an alternative to this optimization of an aggregated loss can be found in partitioning methods [Bragman *et al.* 2019, Maninis *et al.* 2019, Pascal *et al.* 2021a, Strezoski *et al.* 2019a]: by construction, similarly to Maximum Roaming introduced in Chapter 3, these methods optimize each task-specific objective function *alternately and independently*, which is not equivalent to optimizing the average sum of the different task-specific objectives, i.e. the aggregated loss. To the best of my knowledge, only [Maninis *et al.* 2019] discusses it explicitly, as a way to exploit the benefits of partitioning. Therefore, it has not been studied in isolation in any previous work, nor are there theoretical guarantees that this strategy can lead to similar results as the aggregated loss one. This in part can be explained by the fact that the optimization scheme of partitioning methods is always coupled with parameter partitioning, making it difficult to evaluate which part of the contribution is due to the partitioning, and which one is due to the optimization scheme. Moreover, these works do not consider that moving average mechanisms (e.g. momentum) included in state-of-the-art optimizers can mix previous gradient descent directions from different tasks, which means that the individual objective functions are only partially separated.

This chapter studies the task-specific alternate and independent optimization used in partitioning methods and formulates improvements to it by making the following contributions.

- It proves that this optimization strategy provides convergence guarantees similar to the aggregated loss optimization for stochastic gradient descent in

the convex case, and the associated convergence bound is provided.

- It shows that current alternated optimization schemes do not operate truly independent task-specific update steps, due to the momentum mechanisms of most of the existing optimizers, which memorize previous updates, thus bringing them into the current learning steps. I therefore propose a novel alternated optimization scheme that performs truly independent task-specific update steps with momentum-based state-of-the-art optimizers.
- To account for the losses in computational efficiency that alternate independent updates incur into, I introduce a task grouping strategy to reduce training time when dealing with a high number of tasks.
- The proposed optimization strategies consistently lead to substantial improvements in terms of both generalization performance and benchmark results over existing state-of-the-art methods on three well-known benchmark datasets.

## 4.2 Alternate and independent optimization of task-specific objective functions

The standard MTL optimization setup using an aggregated loss is first formalized before the contributions.

### 4.2.1 Standard MTL optimization with aggregated loss

Let  $\xi_t$  be an input data instance randomly sampled during optimization step  $t$ , and  $k$  the index of the  $N$  tasks. The aggregated multi-task objective function to minimize is defined as

$$F(w_t, \xi_t) = \sum_{k=1}^N c^{(k)} \cdot F^{(k)}(w_t, \xi_t), \quad (4.1)$$

where  $F^{(k)}$  is the objective function associated to task  $k$  using shared parameters  $w_t$ , and  $c^{(k)}$  are the task-specific weighting coefficients. For the sake of simplicity in the notation, but without loss of generality, here a uniform weighting of the different task objective functions is considered, i.e.  $c^{(k)} = 1$ . Hereinafter, superscripts are used to index elements associated to different tasks, and subscripts to index the steps of the optimization process.

When using stochastic gradient descent (SGD) to optimize Eq. 4.1, the shared parameters  $w$  at step  $t + 1$  are updated according to the following rule:

$$w_{t+1} = w_t - \eta_t \sum_{k=1}^N \frac{\partial}{\partial w_t} F^{(k)}(w_t, \xi_t), \quad (4.2)$$



## Chapter 4. Separating task-specific objectives for a better optimization

where  $\eta_t$  is the learning rate. For the ease in notation, let us denote  $g^{(k)}(w_t, \xi_t)$  the derivative of  $F^{(k)}$  w.r.t. the parameters  $w_t$ . Eq. 4.2 can be rewritten as

$$w_{t+1} = w_t - \eta_t \sum_{k=1}^N g^{(k)}(w_t, \xi_t). \quad (4.3)$$

In the remaining of the document, this optimization strategy with an aggregated loss is referred as MTL with Shared Update Steps (MTL-SUS) to reflect the characteristics of its update rule (Eq. 4.3).

### 4.2.2 Alternate and independent optimization of task-specific objective functions for SGD

I now formalize the SGD update step for the alternate and independent optimization scheme used in state-of-the-art MTL partitioning works [Bragman *et al.* 2019, Maninis *et al.* 2019, Pascal *et al.* 2021a, Strezoski *et al.* 2019a] and provide convergence bounds in the special case of convex objective functions.

I define a multi-task update step as the alternate Individual Update Steps (IUS) of each of the  $N$  task-specific objective functions, and denote  $w_t^{(k)}$  the parameters optimized w.r.t. task  $k$ , during the  $t$ -th update step. The individual update step  $t + 1$  of task  $k$  is:

$$w_{t+1}^{(k)} = \begin{cases} w_t^{(N)} - \eta_t \cdot g^{(k)}(w_t^{(N)}, \xi_t), & k = 1 \\ w_{t+1}^{(k-1)} - \eta_t \cdot g^{(k)}(w_{t+1}^{(k-1)}, \xi_t), & \forall k > 1. \end{cases} \quad (4.4)$$

To reflect the properties of the update rule (Eq. 4.4), this optimization strategy is denoted as MTL-IUS.

MTL holds analogies with the federated learning problem [Konečný *et al.* 2015, Li *et al.* 2020, Shokri & Shmatikov 2015], where each involved device accounts for one task objective function, with communication after each update step. MTL-SUS corresponds to a full-device participation, and MTL-IUS to a single device participation. Using this analogy, inspiration is taken from [Li *et al.* 2020] to prove that alternate and independent optimization schemes using the MTL-IUS rule converge to the optimum in the case of convex objective functions.

**Assumption 2.**  $F^{(1)}, \dots, F^{(N)}$  are all  $L$ -smooth: for all  $v, w$ ,  $F^{(k)}(v) \leq F^{(k)}(w) + (v - w)^T \nabla F^{(k)}(w) + \frac{L}{2} \|v - w\|_2^2$ .

**Assumption 3.**  $F^{(1)}, \dots, F^{(N)}$  are all  $\mu$ -strongly convex: for all  $v, w$ ,  $F^{(k)}(v) \geq F^{(k)}(w) + (v - w)^T \nabla F^{(k)}(w) + \frac{\mu}{2} \|v - w\|_2^2$ .

**Assumption 4.** Let  $\xi_t$  be uniformly sampled from the data. The variance of stochastic gradients is uniformly bounded:  $\mathbb{E} \|\nabla F^{(k)}(w_t, \xi_t) - \nabla F^{(k)}(w_t)\|^2 \leq \sigma^{(k)^2}$ .

**Assumption 5.** *The expected squared norm of stochastic gradients is uniformly bounded:  $\mathbb{E}[\|\nabla F^{(k)}(w_t, \xi_t)\|^2] \leq G^2$ .*

Let us denote  $F^*$  and  $F^{(k)*}$  as the minima values of  $F$  and  $F^{(k)}$  respectively, and  $w^*$  as the minimizer of  $F$ . Let thus define

$$\Gamma = F^* - \frac{1}{N} \sum_{k=1}^N F^{(k)*},$$

which can be considered as the heterogeneity of the different task-specific objective functions.

**Theorem 1.** *Let  $\gamma \geq 2\frac{L}{\mu} - 1$  and  $\eta_t = \frac{2}{\mu(\gamma+t)}$  the learning rate. Under assumptions 2, 3, 4 and 5, the optimization scheme MTL-IUS presented in Eq. (4.4) satisfies*

$$\mathbb{E}[F(w_T)] - F^* \leq \frac{L}{\gamma + T} \left( \frac{2B}{\mu^2} + \frac{(\gamma + 1)}{2} \mathbb{E}[\|w_1 - w^*\|^2] \right), \quad (4.5)$$

with  $B = \sum_{k=1}^N p_k^2 \sigma_k^2 + 2L\Gamma + G^2$ .

**Proof:** See appendix 4.5  $\square$

Theorem 1 guarantees the convergence of MTL-IUS to the optimum in  $\mathcal{O}(\frac{1}{T})$  with a decreasing learning rate for stochastic gradient descent. However, it should be noted that the bound does not increase directly with the number of tasks, but with the heterogeneity term  $\Gamma$ , which corresponds to the average deviation between the optimum  $F^*$  and the values of the task-specific objective functions  $F^{(k)*}$  at that point.

### 4.2.3 Alternate and independent optimization of task-specific objective functions for moving-average based optimizers

In most of deep learning's literature and in current practice, more complex optimizers than a regular SGD are preferred. State-of-the-art optimizers, such as Adam [Kingma & Ba 2017], generally use exponential moving average mechanisms, e.g. momentum, to smooth and regulate the optimization trajectory. These mechanisms can partially compromise the independence of the task-specific update steps, i.e. MTL-IUS, since every learning step includes descent directions of previous tasks. This section proposes a new update rule that allows to use Individual Optimizers (IO), providing true optimization independence to the different tasks.

#### MTL-IUS with momentum

The update step of MTL-IUS is first reformulated with a momentum-based optimizer, as it is done in partitioning MTL works [Bragman *et al.* 2019, Maninis *et al.* 2019, Pascal *et al.* 2021a].

## Chapter 4. Separating task-specific objectives for a better optimization

Let us define  $\hat{m}$  a generic optimizer rule, which adjusts the gradient descent direction based on exponential moving average mechanisms [Kingma & Ba 2017]. The update step  $t + 1$  of task  $k$  (Eq. (4.4)) can be then reformulated as:

$$w_{t+1}^{(k)} = \begin{cases} w_t^{(N)} - \eta_t \cdot \hat{m} \left( g^{(k)} \left( w_t^{(N)}, \xi_t \right) \right), & k = 1 \\ w_{t+1}^{(k-1)} - \eta_t \cdot \hat{m} \left( g^{(k)} \left( w_{t+1}^{(k-1)}, \xi_t \right) \right), & \forall k > 1, \end{cases} \quad (4.6)$$

where  $\hat{m}$  mixes  $g^{(k)}$  with descent directions  $g^{(j \neq i)}$  of previous tasks update steps. This implies that using this rule does not allow for fully independent task-specific update steps thanks to  $\hat{m}$ .

### Individual Optimizers (MTL-IO)

I propose here to use individual exponential moving averages for each task to allow for fully independent task-specific update steps. I reformulate this update rule, using Individual Optimizers (IOs), as:

$$w_{t+1}^{(k)} = \begin{cases} w_t^{(N)} - \eta_t \cdot \hat{m}^{(k)} \left( g^{(k)} \left( w_t^{(N)}, \xi_t \right) \right), & k = 1 \\ w_{t+1}^{(k-1)} - \eta_t \cdot \hat{m}^{(k)} \left( g^{(k)} \left( w_{t+1}^{(k-1)}, \xi_t \right) \right), & \forall k > 1 \end{cases} \quad (4.7)$$

with  $\hat{m}^{(k)}$  the exponential moving average mechanism of task  $k$ . This alternate and independent optimization strategy is denoted as MTL-IO (Individual Optimizers) to reflect the properties of its update rule. With MTL-IO, the memory term introduced by  $\hat{m}^{(k)}$  only involves previous updates of task  $k$ . This is equivalent to using one IO per task, which avoids the memory leakage that MTL-IUS incurs into.

### 4.2.4 Mitigating computational costs through task grouping

Alternate and independent optimization schemes using MTL-IUS or MTL-IO require one training step per task for each input sample, i.e  $N$  training steps in total. This makes the inference and training time proportional to the number of tasks. To compensate for the increased computational burden, the  $N$  tasks are equally and randomly distributed among a set  $\hat{\mathcal{T}} = \{\hat{\mathcal{T}}_l\}_{l \in \{1..\hat{N}\}}$  of  $\hat{N}$  super-tasks, with  $\hat{N} \in \{1..N\}$ . The set of super-tasks can be optimized with MTL-IUS or MTL-IO, whereas the individual super-task gradients are expressed as the weighted sum of the gradients of their constituting tasks:

$$w_{t+1}^{(l)} = \begin{cases} w_t^{(\hat{N})} - \eta_t \cdot \hat{m} \left( \sum_{k \in \hat{\mathcal{T}}_l} g^{(k)} \left( w_t^{(\hat{N})}, \xi_t \right) \right), & l = 1 \\ w_{t+1}^{(l-1)} - \eta_t \cdot \hat{m} \left( \sum_{k \in \hat{\mathcal{T}}_l} g^{(k)} \left( w_{t+1}^{(l-1)}, \xi_t \right) \right), & \forall l > 1 \end{cases} \quad (4.8)$$

For the sake of brevity, only the formulation adapted for MTL-IUS is presented. Derivation with MTL-IO is straightforward. One should note that this formulation is equivalent to MTL-IUS (respectively, MTL-IO) when  $\hat{N} = T$ , and to MTL-SUS when  $\hat{N} = 1$ . Such task grouping thus defines an adjustable compromise between tasks independence and training speed.

### 4.3 Experiments and results

I report four different experiments performed on three widely used datasets for MTL. The generalization performance of MTL-SUS, MTL-IUS and MTL-IO on NYUv2 [Silberman *et al.* 2012] is first studied, with three heterogeneous tasks, and on the seven homogeneous segmentation tasks of Cityscapes [Cordts *et al.* 2016]. The proposed grouping strategy behavior is then studied on the 40 tasks of Celeb-A [Liu *et al.* 2015b]. Finally are provided insights about the shared parameter space exploration operated by the studied optimization schemes.

In this study, the proposed models are compared with state-of-the-art multi-task methods, including three aggregated loss optimization methods, Grad-Norm [Chen *et al.* 2018], MGDA [Sener & Koltun 2018], PCGrad [Chen *et al.* 2020], and the partitioning method proposed in Chapter 3, Maximum Roaming (MR) [Pascal *et al.* 2021a] using the MTL-IUS optimization scheme. To ensure fairness of comparison, all baselines have been implemented in the same pipeline, under Pytorch 1.2, and run on Nvidia Titan-XP GPUs. For every baseline compared on a dataset, a grid search is performed on the learning rate. All the used scripts to perform these experiments are available on GitHub.<sup>1</sup> The data used comes from official releases for the three datasets.

#### 4.3.1 Scene understanding on NYUv2

The NYUv2 dataset [Silberman *et al.* 2012] provides close to 1500 annotated indoor images extracted from videos captured with the Microsoft Kinect in different buildings. The small number of images and the complexity of the scenes makes it particularly difficult and interesting for multi-task problems. It is used here for one 13-class semantic segmentation task, one depth estimation task, and one normal estimation task. This multi-task setting is judged as particularly challenging, since it is made of three tasks of different natures.

All baselines use a U-Net [Ronneberger *et al.* 2015a], which is known to be particularly efficient for dense labelling tasks. To maximize the proportion of shared weights in the network, all encoders and decoders are shared by all the tasks, while the task-specific prediction heads are made of a single  $1 \times 1$  convolutional layer per task.

The image size is set to  $288 \times 384$  with a batch size of 8. All models are trained for 500 epochs, and all performances are reported over 3 random seeds. Reported metrics are mean Intersection over Union (mIoU) and pixel accuracy for segmentation tasks, absolute and relative error for depth estimation, and mean and median angle error for normal estimation.

<sup>1</sup>[https://github.com/lucas-pascal/AITSO\\_MTL](https://github.com/lucas-pascal/AITSO_MTL).

## Chapter 4. Separating task-specific objectives for a better optimization

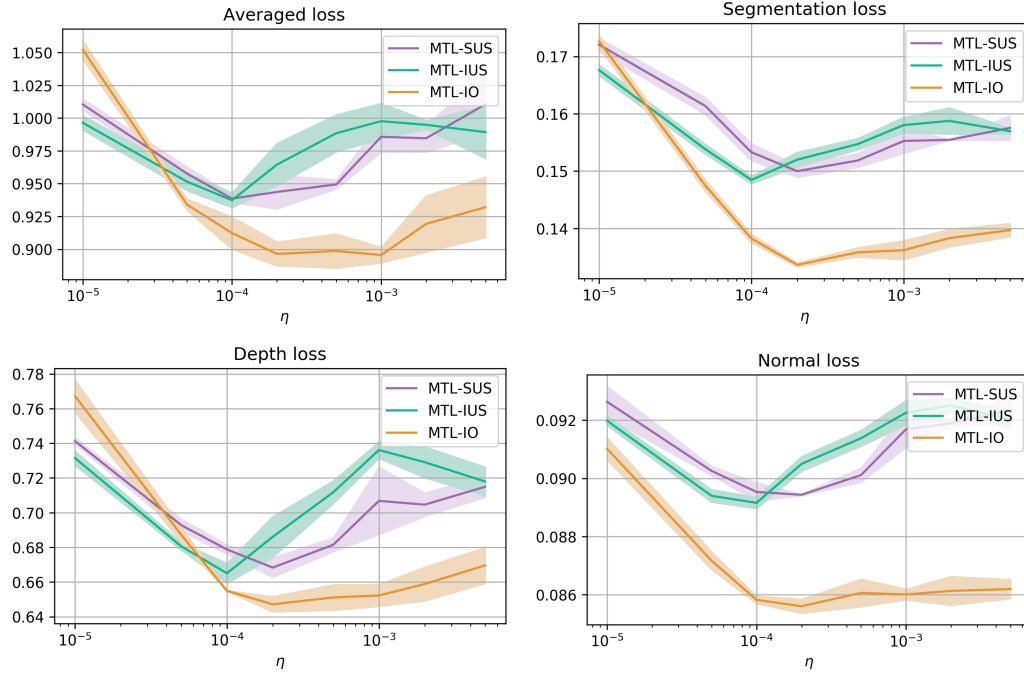


Figure 4.1: NYUv2 best averaged and task-specific validation losses w.r.t the learning rate  $\eta$ .

To understand the general behavior of the compared optimization strategies MTL-SUS, MTL-IUS and MTL-IO, their generalization performance is first studied with respect to the learning rate  $\eta$ , which has a direct influence on the sensitivity of deep networks to minima in the loss landscape, and therefore on their convergence. In Fig. 4.1 are reported the best validation losses achieved by MTL-SUS, MTL-IUS and MTL-IO over a full training, w.r.t. different values of  $\eta$ . It was ensured for every model that the validation loss reached its minimum before the end of the training.

It is first observe that MTL-IO is by far the best overall performing method. Above  $\eta = 5e^{-5}$ , it substantially improves the losses of each of the three tasks, which suggests that it is able to consistently reach better loss regions in the shared parameter space. One can also observe a smaller difference between MTL-IUS and MTL-SUS. Specifically, MTL-IUS performs better for smaller learning rates, and worse beyond  $\eta = 2e^{-4}$ . It is also interesting to note the overall distorted aspect of these curves, with high standard deviations, while more "convex" shapes could be expected. This suggests that the loss landscape is particularly complex, compared to that one of other datasets (see Fig. 4.2,4.3).

The models with the best validation losses for each method are compared with the state-of-the-art baselines (Table 4.1). While MTL-IUS and MTL-SUS perform similarly, MTL-IO achieves the best results overall on segmentation and depth

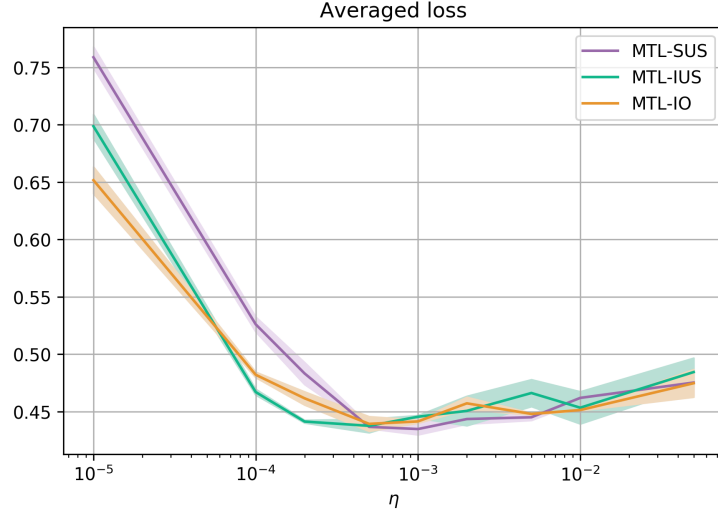


Figure 4.2: Cityscapes best averaged and task-specific validation losses w.r.t the learning rate  $\eta$ .

estimation metrics, and falls close behind MR for normals estimation, making it the best performing multi-task method on this dataset.

### 4.3.2 Multi-class segmentation on Cityscapes

The Cityscapes dataset [Cordts *et al.* 2016] provides 5K fine-grained annotated street-view images captured from a car point of view, in many different German cities. The 7 semantic segmentation classes are used here as 7 independent tasks, to observe how the different optimization methods behave with more homogeneous tasks. The same network as for NYUv2 is used, and images are resized to  $128 \times 256$  with a batch size of 24. A similar study to the one conducted on NYUv2 is reported in Fig. 4.2, and a benchmark in Table 4.2.

Here can be observed smoother curves with smaller variations, which confirms the intuition that this setting is easier to optimize. The three optimization methods report very similar results. While the best overall validation loss is reached by MTL-SUS, MTL-IUS and MTL-IO are consistently better in the lower learning rates, suggesting that they are more robust to local minima in the loss landscape. As one could expect given Fig. 4.2, all the methods, including the state-of-the-art benchmarks, present a very close performance. It seems like in a simple setting with a well calibrated learning rate, the influence of a particular optimization method is greatly reduced.

## Chapter 4. Separating task-specific objectives for a better optimization

	Segmentation		Depth estimation		Normals estimation		Rank (↓)
	mIoU (↑)	Pix. Acc. (↑)	Abs. Err. (↓)	Rel. Err. (↓)	Mean Err. (↓)	Med. Err. (↓)	
MTL-SUS	24.29 (±0.38)	48.79 (±0.35)	68.48 (±0.28)	52.93 (±0.59)	27.28 (±0.02)	33.70 (±0.03)	4.33
MTL-IUS	24.93 (±0.33)	48.91 (±0.17)	67.76 (±0.61)	53.25 (±0.77)	27.28 (±0.05)	33.70 (±0.05)	3.88
MTL-IO	<b>29.48</b> (±0.30)	<b>54.06</b> (±0.21)	<b>66.07</b> (±0.68)	<b>50.27</b> (±0.28)	<b>27.08</b> (±0.03)	<b>33.35</b> (±0.05)	<b>1.00</b>
GradNorm	28.09 (±0.85)	52.91 (±0.74)	70.26 (±1.4)	52.40 (±1.35)	27.18 (±0.05)	33.44 (±0.07)	3.16
PCGrad	24.51 (±0.32)	48.87 (±0.41)	67.89 (±0.67)	52.29 (±0.75)	27.32 (±0.04)	33.65 (±0.08)	3.50
MR	28.82 (±0.18)	<b>53.77</b> (±0.61)	71.30 (±0.97)	53.37 (±0.97)	<b>27.03</b> (±0.02)	<b>33.27</b> (±0.03)	2.66

Table 4.1: NYUv2 results with standard deviations. The best results (and statistically equivalent) per column are highlighted in each category. Methods ranks are computed by averaging the rank of the methods on each metric.

### 4.3.3 Multi-attribute segmentation on Celeb-A

The CelebA dataset [Liu *et al.* 2015b] contains 200K face images of 10K different celebrities. Annotations are provided for 40 different facial attributes. In this experiment is studied the proposed random grouping strategy with a large number of tasks, to establish to which extent it can reduce the computational overhead of MTL-IUS and MTL-IO without affecting their performance. Each of the 40 provided facial attributes is considered as an independent facial attribute detection task, and different groupings are operated on them,  $\hat{N} = \{2, 4, 8, 20, 40\}$ .

A shallow 9-layer fully convolutional network similar to [Chen *et al.* 2020] is used, with a task-specific fully-connected prediction layer for each task (or task group). The image size is set to  $64 \times 64$ , with a batch size of 256. All models are trained over 15 epochs. The average classification error is reported along with F1-score, since the different attributes are not balanced. Reported results are averaged over 3 seeds, with new task groups are sampled for every seed. Figure 4.3 shows the best achieved validation losses over training w.r.t. the learning rate and for different number of super-tasks  $\hat{N}$ . Table 4.3 summarizes the benchmark result.

For  $\hat{N} \leq 8$ , MTL-IUS and MTL-IO are able to reach an improvement compared to MTL-SUS. Then, the performance decreases up to  $\hat{N} = 40$  (i.e. no grouping). This suggests that the use of a grouping strategy is beneficial both in terms of

	Segmentation		
	mIoU ( $\uparrow$ )	Pix. Acc. ( $\uparrow$ )	Rank ( $\downarrow$ )
MTL-SUS	<b>69.79 <math>\pm</math> 0.40</b>	<b>90.40 <math>\pm</math> 0.08</b>	<b>1.0</b>
MTL-IUS	<b>69.74 <math>\pm</math> 0.05</b>	90.22 $\pm$ 0.13	2.5
MTL-IO	69.52 $\pm$ 0.26	<b>90.28 <math>\pm</math> 0.14</b>	2.5
GradNorm	69.00 $\pm$ 0.12	90.17 $\pm$ 0.10	5.0
PCGrad	<b>69.73 <math>\pm</math> 0.05</b>	<b>90.42 <math>\pm</math> 0.06</b>	<b>1.0</b>
MR	69.15 $\pm$ 0.08	89.79 $\pm$ 0.08	5.5

Table 4.2: Cityscape results with standard deviations. The best results (and statistically equivalent) per column are highlighted. Methods ranks are computed by averaging the rank of the methods on each metric.

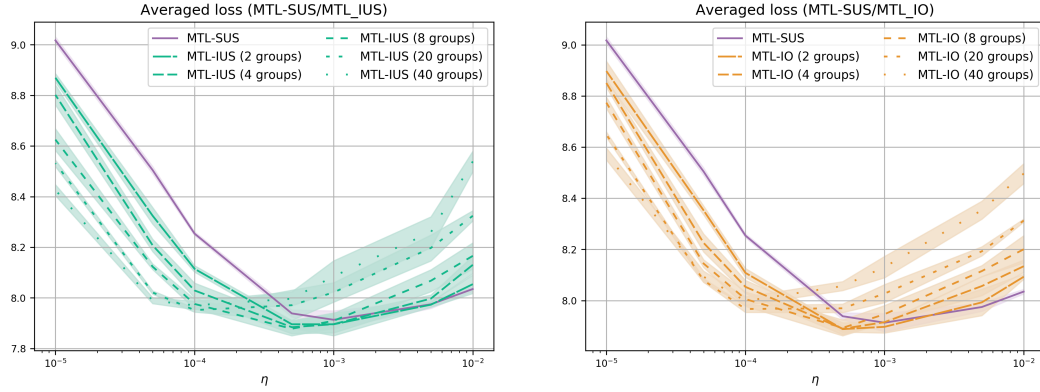


Figure 4.3: Celeb-A best averaged validation losses w.r.t the learning rate  $\eta$  and different numbers of task groups. **(Left)** MTL-IUS models compared to MTL-SUS. **(Right)** MTL-IO models compared to MTL-SUS.

performance and computational efficiency when dealing with a large number of tasks. The latter is further confirmed in Table 4.3, where the best grouping models of MTL-IUS and MTL-IO also outperform other state-of-the-art methods on both average error and F-score metrics.

More generally, a progressive evolution of the curves w.r.t. the different values of  $\hat{N}$  can be observed: as with Cityscapes, the less tasks are grouped together, the better is the optimization of IUS and IO in the low learning rates, while it is the opposite for high learning rates. This behavior suggests that these methods introduce more noise in the optimization, making them less sensitive to local minima, and pushing for a larger exploration of the shared parameter space.



## Chapter 4. Separating task-specific objectives for a better optimization

	Avg. error ( $\downarrow$ )	F1-score ( $\uparrow$ )	Rank ( $\downarrow$ )
MTL-SUS	$9.20 \pm 0.02$	$68.66 \pm 0.24$	6.0
MTL-IUS (2 groups)	$9.16 \pm 0.03$	$68.69 \pm 0.21$	—
MTL-IUS (4 groups)	$9.13 \pm 0.02$	$69.01 \pm 0.27$	—
MTL-IUS (8 groups)	<b><math>9.10 \pm 0.03</math></b>	$68.93 \pm 0.38$	3.0
MTL-IUS (20 groups)	$9.20 \pm 0.05$	$68.82 \pm 0.11$	—
MTL-IUS (40 groups)	$9.27 \pm 0.05$	$68.97 \pm 0.17$	—
MTL-IO (2 groups)	$9.15 \pm 0.02$	$68.85 \pm 0.33$	—
MTL-IO (4 groups)	<b><math>9.11 \pm 0.02</math></b>	<b><math>69.48 \pm 0.15</math></b>	<b>1.0</b>
MTL-IO (8 groups)	$9.15 \pm 0.01$	$68.55 \pm 0.09$	—
MTL-IO (20 groups)	$9.23 \pm 0.02$	$68.61 \pm 0.31$	—
MTL-IO (40 groups)	$9.24 \pm 0.00$	$68.57 \pm 0.04$	—
GradNorm	$9.16 \pm 0.03$	$69.16 \pm 0.19$	4.0
PCGrad	$9.16 \pm 0.04$	<b><math>69.26 \pm 0.30</math></b>	2.5
MGDA-UB	$9.42 \pm 0.03$	$67.68 \pm 0.12$	7.0
MR	<b><math>9.12 \pm 0.00</math></b>	<b><math>69.21 \pm 0.31</math></b>	<b>1.0</b>

Table 4.3: Celeb-A results with standard deviations. The best results (and statistically equivalent) per column are highlighted. In parenthesis are reported the different number of task groups for MTL-IUS and IO. The best (4 and 8) and worst (40) performing numbers of task groups are reported for both the strategies.

### 4.3.4 Covered distance

Here are provided insights about how the studied optimization strategies explore the shared parameter space, by measuring the distances covered in this space during training. Using the Frobenius norm, measures are provided for the shortest path from the network’s initialization to the loss minimum (shortest) and the total covered distance (total), which is the sum of the distances covered at each update step. The total-to-shortest-distance ratio is also reported, which gives an insight on how much the surrounding space has been explored for a given shortest path.

Table 4.4 summarizes the obtained measures during the optimization of MTL-SUS, MTL-IUS and MTL-IO over all datasets. All three baselines start from the same parameter initialization (i.e. same point in the parameter space) and use the same learning rate (the one providing the best performances overall). The measurements are performed at the same point in time (the closest from the validation loss minimum). The results are averaged over 3 different seeds. As a reminder, the aggregated multi-task loss uses uniform weights of 1 for every task, so that the gradient scaling is the same for the three optimization strategies.

It can be observed that both MTL-IUS and MTL-IO travel more, both in terms

Dataset	Model	Total covered dist.	Shortest path	Ratio
NYUv2	SUS	$879.33 \pm 18.22$	$38.77 \pm 0.53$	$22.68 \pm 0.17$
	IUS	$2597.74 \pm 49.59$	$67.44 \pm 1.02$	$38.52 \pm 0.24$
	IO	$2769.52 \pm 40.20$	$71.06 \pm 0.80$	$38.97 \pm 0.22$
Cityscapes	SUS	$909.38 \pm 10.02$	$74.38 \pm 0.80$	$12.23 \pm 0.01$
	IUS	$8102.39 \pm 13.16$	$262.68 \pm 0.48$	$30.85 \pm 0.02$
	IO	$8438.78 \pm 105.47$	$277.29 \pm 5.69$	$30.44 \pm 0.35$
Celeb-A	SUS	$1398.01 \pm 3.54$	$99.66 \pm 0.22$	$14.03 \pm 0.00$
	IUS (4 groups)	$5636.18 \pm 19.59$	$223.83 \pm 0.69$	$25.18 \pm 0.02$
	IO (4 groups)	$6560.56 \pm 45.21$	$242.55 \pm 1.80$	$27.05 \pm 0.12$
	IUS (8 groups)	$14955.87 \pm 15.55$	$387.13 \pm 0.48$	$38.63 \pm 0.01$
	IO (8 groups)	$13992.50 \pm 268.21$	$369.67 \pm 5.39$	$37.85 \pm 0.17$
	IUS (40 groups)	$65376.36 \pm 164.54$	$837.32 \pm 1.91$	$78.08 \pm 0.08$
	IO (40 groups)	$59627.39 \pm 746.35$	$764.42 \pm 9.01$	$78.00 \pm 0.07$

Table 4.4: Total covered and shortest path distances in the shared parameter space from the same initialization point.

of total covered and shortest path distances. The distances increase globally with the number of tasks (or task groups). The increase in the shortest path distance indicates that these methods are able to discover more distant loss regions compared to MTL-SUS. These observations correlate with the findings from the previous experiments, suggesting that some of these more distant regions can provide better generalization performance.

Most interestingly, MTL-IUS and MTL-IO present a higher total-to-shortest path distance ratio compared to MTL-SUS, which means that between two locations in the shared parameter space, MTL-IUS and MTL-IO follow a much more oscillating trajectory. This observation suggests that optimizing tasks-specific objective functions, instead of an aggregated multi-task objective one, introduces more stochasticity to the optimization, which is usually beneficial in deep learning.

## 4.4 Discussion

This Chapter studied the *alternate and independent* optimization of the task-specific objective functions (MTL-IUS) used in partitioning methods, and compared it to the optimization of the more standard aggregated objective function (MTL-SUS), which is widely adopted in the literature. Taking inspiration from Federated Learning works [Konečný *et al.* 2015, Li *et al.* 2020], it was first demonstrated that the former has similar convergence properties as the latter when dealing with convex objective functions. It was then showed that the existing partitioning

## Chapter 4. Separating task-specific objectives for a better optimization

---

methods do not operate truly independent update steps due to the momentum mechanisms included in state-of-the-art optimizers. A novel strategy was thus formulated, using an independent optimizer per task (MTL-IO), that favors task independence. To account for the computational overhead of these strategies, a random grouping strategy was proposed, striking a balance between computational efficiency, and the benefits of a task-independent optimization strategy.

The experimental results over three datasets show both MTL-IUS and MTL-IO achieve an overall better generalization performance w.r.t standard aggregated objective optimization (MTL-SUS) and state-of-the-art MTL baselines using an aggregated objective function. In particular, MTL-IO shows important improvements in settings where tasks are of a very different nature. The results also confirm that the proposed random grouping strategy applied to MTL-IUS and MTL-IO is beneficial both in terms of performance and efficiency, when dealing with a great number of tasks. Finally, it was showed that MTL-IUS and IO allow parameters updates to travel a longer distance, and ensure a more thorough exploration of the shared parameter space.

While the optimization of an aggregated multi-task objective function is widely adopted in the literature, the presented investigations suggest that its usage should be questioned, despite the computational benefits it provides.

### 4.5 Appendix

In this section is proved the convergence of MTL-IUS for stochastic gradient descent when dealing with convex objective functions (Theorem 1). Inspiration is drawn from the proof of convergence of the FedAvg algorithm [Li *et al.* 2020] for partial device participation in federated learning.

#### Problem setting and notations

For the ease of notation, in the following, the double indexing of  $w$  (tasks and iterations) used in equation 4.4 is dropped. The IUS update rule (i.e. equation 4.4) of the weights at update step  $t + 1$  thus gets re-written as:

$$\begin{cases} v_{t+1}^k = w_t - \eta_t \nabla F^{(k)}(w_t, \xi_t^k), \\ w_{t+1} = v_{t+1}^{s_t}, \end{cases}$$

where  $v_{t+1}^k$  is the SGD from  $w_t$  with respect to task  $k$  objective function, and  $w_{t+1}$  only follows the SGD of one task  $s_t$  (while others are not considered), with  $s_t$  selected in  $\{1, \dots, N\}$ , for every step  $t$ . The theorem is proved for  $s_t$  randomly selected at uniform at each iteration  $t$ , which also proves it for the case proposed here, since the proof only assumes a sampling probability of  $p_k = \frac{1}{N}$  for any task  $k$ .

Let us define  $\bar{v}_t = \frac{1}{N} \sum_{k=1}^N v_t^k$  the averaged SGD at step  $t+1$ ,  $\bar{g}_t = \frac{1}{N} \sum_{k=1}^N \nabla F^{(k)}(w_t)$  the averaged gradient and  $g_t = \frac{1}{N} \sum_{k=1}^N \nabla F^{(k)}(w_t, \xi_t^k)$  the averaged stochastic gradient. This gives  $\bar{v}_{t+1} = w_t - \eta_t g_t$  and  $\mathbb{E}g_t = \bar{g}_t$ .

### Key lemmas

Two lemmas used in the theorem's proof are first stated. Their proof is left for Appendix 4.5.

**Lemma 3.** *Under assumptions 2 and 3, if  $\eta_t \leq \frac{1}{L}$ , one can obtain:*

$$\mathbb{E}||\bar{v}_{t+1} - w^*||^2 \leq (1 - \mu\eta_t)\mathbb{E}||w_t - w^*||^2 + 2L\eta_t^2\Gamma + \frac{\eta_t^2}{N^2} \sum_{k=1}^N \sigma_k^2 \quad (4.9)$$

**Lemma 4.** *For  $\eta_t$  non-increasing such as  $\eta_t \leq 2\eta_{t+1}, \forall t \geq 0$ , one can obtain:*

$$\mathbb{E}_{s_t}||\bar{v}_{t+1} - w_{t+1}||^2 \leq \eta_t^2 G^2 \quad (4.10)$$

### Theorem proof

$$||w_{t+1} - w^*||^2 = \underbrace{||w_{t+1} - \bar{v}_{t+1}||^2}_{A_1} + \underbrace{||\bar{v}_{t+1} - w^*||^2}_{A_2} + \underbrace{2\langle w_{t+1} - \bar{v}_{t+1}, \bar{v}_{t+1} - w^* \rangle}_{A_3}$$

For  $A_3$ , one can obtain

$$\mathbb{E}_{s_t}(w_{t+1}) = \mathbb{E}_{s_t}(v_{t+1}^{s_t}) = \frac{1}{N} \sum_{k=1}^N v_{t+1}^k = \bar{v}_{t+1}$$

which proves  $\mathbb{E}_{s_{t+1}}(A_3) = 0$ , where  $\mathbb{E}_{s_t}$  accounts for the expectation taken over the random sampling of  $s_t$ .

$A_1$  is bounded with Lemma 2 and  $A_2$  with Lemma 1:

$$\mathbb{E}||w_{t+1} - w^*||^2 \leq (1 - \mu\eta_t)\mathbb{E}||w_t - w^*||^2 + \frac{\eta_t^2}{N^2} \sum_{k=1}^N \sigma_k^2 + 2L\eta_t^2\Gamma + \eta_t^2 G^2$$

Let  $\Delta_t = \mathbb{E}||w_{t+1} - w^*||^2$  and  $B = \frac{1}{N^2} \sum_{k=1}^N \sigma_k^2 + 2L\Gamma + G^2$ . This gives:

$$\Delta_{t+1} \leq (1 - \mu\eta_t)\Delta_t + \eta_t^2 B$$

One can show by induction that for  $\beta > \frac{1}{\mu}$  and  $\gamma > 0$  verifying  $\eta_1 \leq \frac{1}{L}$  and  $\eta_t \leq 2\eta_{t+1}$ , one obtains  $\Delta_t \leq \frac{v}{\gamma+t}$  with  $v = \max\{\frac{\beta^2 B}{\beta\mu-1}, (\gamma+1)\Delta_1\}$ :

## Chapter 4. Separating task-specific objectives for a better optimization

The affirmation directly holds for  $t = 1$ . Then, suppose it holds for  $t$ , this gives:

$$\begin{aligned}
\Delta_{t+1} &\leq (1 - \mu\eta_t)\Delta_t + \eta_t^2 B \\
&\leq \left(1 - \frac{\beta\mu}{t+\gamma}\right) \frac{v}{t+\gamma} + \frac{\beta^2 B}{(t+\gamma)^2} \\
&= \frac{t+\gamma-1}{(t+\gamma)^2} v + \left[ \frac{\beta^2 B}{(t+\gamma)^2} - \frac{\beta\mu-1}{(t+\gamma)^2} v \right] \\
&\leq \frac{t+\gamma-1}{(t+\gamma)^2} v \\
&\leq \frac{t+\gamma-1}{(t+\gamma+1)(t+\gamma-1)} v = \frac{v}{(t+1)+\gamma}
\end{aligned}$$

where the second inequality comes from  $v \geq \frac{\beta^2 B}{\beta\mu-1}$ . This proves that the affirmation holds for  $t+1$ .

Which gives, with the convexity of  $F$ :

$$\mathbb{E}[F(w_t)] - F^* \leq \frac{L}{2(\gamma+t)} \left( \frac{\beta^2 B}{\beta\mu-1} + (\gamma+1) \|w_1 - w^*\|^2 \right)$$

which proves theorem 1 for  $\beta = \frac{2}{\mu}$ , and  $\gamma \geq 2\frac{L}{\mu} - 1$ , to ensure  $\eta_1 \leq \frac{1}{L}$ .

### Proof of key lemmas

**Lemma 1** Let assumptions 2 and 3 hold, and find an upper bound for  $\mathbb{E} \|\bar{v}_{t+1} - w^*\|^2$ :

$$\begin{aligned}
\|\bar{v}_{t+1} - w^*\|^2 &= \|w_t - \eta_t g_t - w^* - \eta_t \bar{g}_t + \eta_t \bar{g}_t\|^2 \\
&= \underbrace{\|w_t - w^* - \eta_t \bar{g}_t\|^2}_{A_1} + \underbrace{2\eta_t \langle w_t - w^* - \eta_t \bar{g}_t, \bar{g}_t - g_t \rangle}_{A_2} + \eta_t^2 \|g_t - \bar{g}_t\|^2
\end{aligned} \tag{4.11}$$

Note that  $\mathbb{E}[A_2] = 0$ , so the focus gets to  $A_1$ :

$$\begin{aligned}
A_1 &= \|w_t - w^* - \eta_t \bar{g}_t\|^2 \\
&= \|w_t - w^*\|^2 - \underbrace{2\eta_t \langle w_t - w^*, \bar{g}_t \rangle}_{B_1} + \underbrace{\eta_t^2 \|\bar{g}_t\|^2}_{B_2}
\end{aligned}$$

By convexity of  $\|\cdot\|^2$  and  $L$ -smoothness of  $F^{(k)}(\cdot)$ , one gets:

$$\begin{aligned} B_2 &= \eta_t^2 \|\bar{g}_t\|^2 \\ &\leq \frac{\eta_t^2}{N} \sum_{k=1}^N \|\nabla F^{(k)}(w_t)\|^2 \\ &\leq 2L \frac{\eta_t^2}{N} \sum_{k=1}^N (F^{(k)}(w_t) - F^{(k)*}) \end{aligned}$$

Then:

$$\begin{aligned} B_1 &= -2\eta_t \langle w_t - w^*, \bar{g}_t \rangle \\ &= -2 \frac{\eta_t}{N} \sum_{k=1}^N \langle w_t - w^*, \nabla F^{(k)}(w_t) \rangle \end{aligned}$$

The  $\mu$ -strong convexity of  $F^{(k)}(\cdot)$  gives:

$$- \langle w_t - w^*, \nabla F^{(k)}(w_t) \rangle \leq -(F^{(k)}(w_t) - F^{(k)}(w^*)) - \frac{\mu}{2} \|w_t - w^*\|^2$$

it is then replaced in  $A_1$ :

$$\begin{aligned} A_1 &= \|w_t - w^* - \eta_t \bar{g}_t\|^2 \\ &\leq \|w_t - w^*\|^2 + 2L \frac{\eta_t^2}{N} \sum_{k=1}^N (F^{(k)}(w_t) - F^{(k)*}) \\ &\quad - 2 \frac{\eta_t}{N} \sum_{k=1}^N \left( (F^{(k)}(w_t) - F^{(k)}(w^*)) + \frac{\mu}{2} \|w_t - w^*\|^2 \right) \\ &= (1 - \mu\eta_t) \|w_t - w^*\|^2 \\ &\quad + \underbrace{2L \frac{\eta_t^2}{N} \sum_{k=1}^N (F^{(k)}(w_t) - F^{(k)*}) - 2 \frac{\eta_t}{N} \sum_{k=1}^N (F^{(k)}(w_t) - F^{(k)}(w^*))}_C \end{aligned}$$

## Chapter 4. Separating task-specific objectives for a better optimization

Let  $\gamma_t = 2\eta_t(1 - L\eta_t)$ . Since  $\eta_t \leq \frac{1}{L}$ , one obtains  $\gamma_t \geq 0$ . Then:

$$\begin{aligned}
C &= 2L\frac{\eta_t^2}{N} \sum_{k=1}^N (F^{(k)}(w_t) - F^{(k)*}) - 2\frac{\eta_t}{N} \sum_{k=1}^N (F^{(k)}(w_t) - F^{(k)}(w^*)) \\
&= 2L\frac{\eta_t^2}{N} \sum_{k=1}^N (F^{(k)}(w_t) - F^{(k)*}) - 2\frac{\eta_t}{N} \sum_{k=1}^N (F^{(k)}(w_t) + F^{(k)*} - F^{(k)*} - F^{(k)}(w^*)) \\
&= \left[ \frac{1}{N} \sum_{k=1}^N (F^{(k)}(w_t) - F^{(k)*}) \right] (2L\eta_t^2 - 2\eta_t) + 2\frac{\eta_t}{N} \sum_{k=1}^N (F^{(k)}(w^*) - F^{(k)*}) \\
&= -\frac{\gamma_t}{N} \sum_{k=1}^N (F^{(k)}(w_t) - F^{(k)*}) + 2\frac{\eta_t}{N} \sum_{k=1}^N (F^{(k)}(w^*) - F^{(k)*}) \\
&= -\frac{\gamma_t}{N} \sum_{k=1}^N (F^{(k)}(w_t) - F^* + F^* - F^{(k)*}) + 2\frac{\eta_t}{N} \sum_{k=1}^N (F^* - F^{(k)*}) \\
&= -\frac{\gamma_t}{N} \sum_{k=1}^N (F^{(k)}(w_t) - F^*) + \frac{2\eta_t - \gamma_t}{N} \sum_{k=1}^N (F^* - F^{(k)*}) \\
&= -\frac{\gamma_t}{N} \sum_{k=1}^N (F^{(k)}(w_t) - F^*) + 2L\frac{\eta_t^2}{N} \sum_{k=1}^N (F^* - F^{(k)*}) \\
&= -\frac{\gamma_t}{N} \sum_{k=1}^N (F^{(k)}(w_t) - F^*) + 2L\eta_t^2 \Gamma \\
&= -\gamma_t (F(w_t) - F^*) + 2L\eta_t^2 \Gamma \leq 2L\eta_t^2 \Gamma
\end{aligned}$$

Since  $(F(w_t) - F^*) \geq 0$  and  $\gamma_t \geq 0$ . It is then replaced in equation 4.11:

$$\begin{aligned}
\|\bar{v}_{t+1} - w^*\|^2 &\leq (1 - \mu\eta_t)\|w_t - w^*\|^2 + 2L\eta_t^2 \Gamma + \eta_t^2 \|g_t - \bar{g}_t\|^2 \\
\mathbb{E}\|\bar{v}_{t+1} - w^*\|^2 &\leq (1 - \mu\eta_t)\mathbb{E}\|w_t - w^*\|^2 + 2L\eta_t^2 \Gamma + \eta_t^2 \mathbb{E}\|g_t - \bar{g}_t\|^2
\end{aligned}$$

Then, Assumption 3 gives us:

$$\begin{aligned}
\mathbb{E}\|g_t - \bar{g}_t\|^2 &= \mathbb{E}\left\| \frac{1}{N} \sum_{k=1}^N (\nabla F^{(k)}(w_t, \xi_t^k) - \nabla F^{(k)}(w_t)) \right\|^2 \\
&= \frac{1}{N^2} \sum_{k=1}^N \mathbb{E}\|\nabla F^{(k)}(w_t, \xi_t^k) - \nabla F^{(k)}(w_t)\|^2 \\
&\leq \frac{1}{N^2} \sum_{k=1}^N \sigma_k^2
\end{aligned}$$

One finally obtains:

$$\mathbb{E} \|\bar{v}_{t+1} - w^*\|^2 \leq (1 - \mu\eta_t) \mathbb{E} \|w_t - w^*\|^2 + 2L\eta_t^2\Gamma + \frac{\eta_t^2}{N^2} \sum_{k=1}^N \sigma_k^2$$

**Lemma 2**

Now is to find an upper bound for  $\mathbb{E}_{s_t} \|\bar{v}_{t+1} - w_{t+1}\|^2$ , with  $\mathbb{E}_{s_t}$  the expected value taken over  $s_t$  values:

$$\begin{aligned} \mathbb{E}_{s_t} \|\bar{v}_{t+1} - w_{t+1}\|^2 &= \mathbb{E}_{s_t} \|v_{t+1}^{s_t} - \bar{v}_{t+1}\|^2 \\ &= \frac{1}{N} \sum_{k=1}^N \|v_{t+1}^k - \bar{v}_{t+1}\|^2 \\ &= \frac{1}{N} \sum_{k=1}^N \|(v_{t+1}^k - w_t) - (\bar{v}_{t+1} - w_t)\|^2 \\ &= \mathbb{E} \|(v_{t+1}^k - w_t) - \mathbb{E}(v_{t+1}^k - w_t)\|^2 \end{aligned}$$

Since  $\mathbb{E} \|x - \mathbb{E}x\|^2 \leq \mathbb{E} \|x\|^2$ , one obtains:

$$\begin{aligned} \mathbb{E}_{s_t} \|\bar{v}_{t+1} - w_{t+1}\|^2 &\leq \mathbb{E} \|v_{t+1}^k - w_t\|^2 = \frac{1}{N} \sum_{k=1}^N \|v_{t+1}^k - w_t\|^2 \\ &\leq \frac{1}{N} \sum_{k=1}^N \mathbb{E} \|v_{t+1}^k - w_t\|^2 \\ &\leq \frac{1}{N} \sum_{k=1}^N \mathbb{E} \|\eta_t \nabla F^{(k)}(w_t, \xi_t^k)\|^2 \\ &\leq \frac{1}{N} \sum_{k=1}^N \eta_t^2 G^2 = \eta_t^2 G^2 \end{aligned}$$

which proves the Lemma.





# Glaucoma Diagnosis from Retinal Fundus Imaging through MTL

---

## Contents

---

<b>5.1 Motivation</b>	<b>61</b>
5.1.1 Deep multi-task networks for automated glaucoma diagnosing	61
5.1.2 The Retinal Fundus Imaging challenge (REFUGE)	63
<b>5.2 Related work</b>	<b>64</b>
<b>5.3 Method</b>	<b>64</b>
5.3.1 Pipeline description	65
5.3.2 Losses and metrics	67
5.3.3 Optimization	68
<b>5.4 Experiments and Results</b>	<b>68</b>
5.4.1 Experimental details	68
5.4.2 Experimental Results	69
5.4.3 Combination with Transfer Learning	73
<b>5.5 Discussion</b>	<b>76</b>

---

This chapter is adapted from "Detection, Segmentation and localization using a single model in Glaucoma detection from color fundus images", a work presented at the second edition of the Retinal Fundus Glaucoma challenge (REFUGE2) <sup>1</sup>, organized at the MICCAI 2020 conference.

## 5.1 Motivation

### 5.1.1 Deep multi-task networks for automated glaucoma diagnosing

Glaucoma is a prevalent condition related to an abnormal fluid balance in the eye that causes an increase of internal ocular pressure. The increase of pressure gradually damages the eye optic nerve. If not diagnosed, these damages can result in permanent vision loss. Approximately 4% of the global population between 40 and 80 years old is affected by glaucoma [Tham *et al.* 2014]. Patients affected by

---

<sup>1</sup><https://refuge.grand-challenge.org/Home2020/>

glaucoma usually do not present symptoms in the early stages of the disease, while an early diagnosis is critical to prevent irreversible damages. It is thus important to develop inexpensive detection methods, in order to massively and systematically control patients, before the symptoms appear.

One way to diagnose glaucoma is to perform a visual examination of the inside back surface of the eye, named fundus. The images are obtained by special cameras through a dilated pupil. An example of such imaging technique is shown in Figures 5.2 (left) and 5.1. The main advantage of this technique, called "Retinal Fundus Imaging", is to be brief and painless to the patient, making it suited for simple routine checks. However, establishing an accurate diagnosis from these images is particularly difficult, and generally requires human experts. The most discriminant symptom for detecting glaucoma on fundus images is the presence of a "cupping", which is retraction of the optic disc on the optic cup. This cupping causes an increase of the vertical Cup-to-Disc ratio, which is the height ratio between the optic cup (OC) and optic disc (OD). Other markers of glaucoma exist, more difficult to detect, such as peripapillary hemorrhages and Retinal nerve fiber layer defects [Orlando *et al.* 2020].

While diagnosing campaigns are regularly launched, these require human experts to examine large amounts of samples in limited time [Orlando *et al.* 2020]. Therefore, automatic glaucoma detection methods are desirable. Deep convolutional networks have shown beneficial for years in medical imaging tasks [Tajbakhsh *et al.* 2016], and seem particularly adapted for such task, only involving the identification of some specific visual patterns in images. Since deep networks learn their own set of discriminant features, these models might also leverage other relevant features than just the vCDR. However, automating glaucoma diagnosis with deep networks essentially suffers from two limitations:

- The lack of data: existing annotated datasets contain at most few hundreds of samples [Hagiwara *et al.* 2018].
- The disparity of data: the data can vary a lot depending on the imaging device involved and its calibration. One solution developed in one particular medical center may perform poorly in another medical center [Orlando *et al.* 2020].

These issues are highly problematic for training deep neural networks, which demand large amounts of data to reach correct generalization performance. Therefore, a Multi-Task Learning approach is of particular relevance in this context, to maximize the benefits of the few annotated data at disposal and create better generalization performances. Although most existing works on glaucoma diagnosis use single-task approaches [Hagiwara *et al.* 2018, Orlando *et al.* 2020], this chapter proposes to use a multi-task one. As a result, only a single model is required for the different tasks involved in glaucoma diagnosis. The sharing and optimization methods proposed in previous chapters are used, to mitigate possible task interference and improve

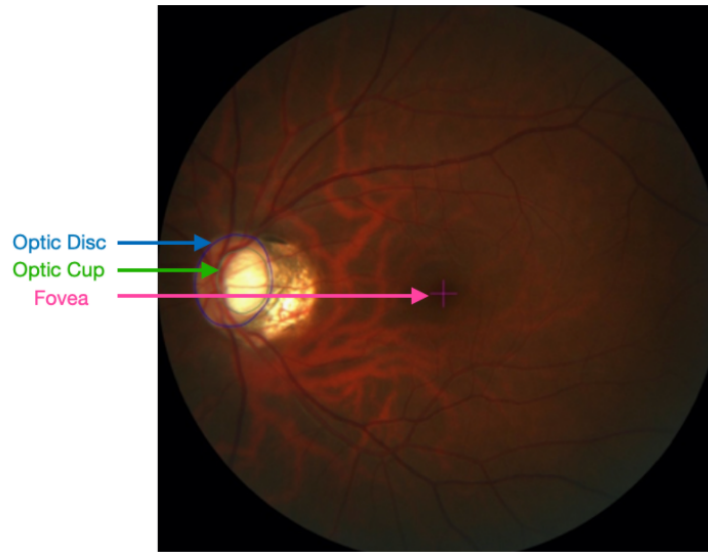


Figure 5.1: Example of a retinal fundus image, with annotations for Optic Disc, Optic Cup and Fovea. (Source: [Orlando *et al.* 2020])

further the generalization performance in a concrete application, such as glaucoma diagnosis.

### 5.1.2 The Retinal Fundus Imaging challenge (REFUGE)

In 2018 has been organized the “Retinal Fundus Glaucoma Challenge” (REFUGE), included as a satellite event in the MICCAI conference (2018). For this event, 1200 retinal fundus images (400 for training, 400 for validation, 400 for testing) from different cameras and medical centers have been collected and annotated by human experts. Annotations were provided for four different tasks:

- **Glaucoma diagnosis:** binary labels, attesting for the presence of glaucoma.
- **Optic Disc segmentation map:** The optic disc is the region defined by the optic nerve head.
- **Optic Cup segmentation map:** The optic cup is the white elliptic region located inside the optic disc.
- **Fovea localization:** The fovea is the sharp central vision point.

A visualization of such annotations is provided in Figure 5.1.

In the following, the data provided in this challenge is used to experiment the methods proposed in this thesis in a multi-task pipeline, with the aim of improving the generalization performance of deep networks in a real world application.

## 5.2 Related work

In [Orlando *et al.* 2020] it is presented a report of the first edition of the REFUGE challenge, along with thorough description of the methods submitted by the 12 best ranked contenders. Note however that the fovea localization tasks was not part of the contest in this first edition. All the reported strategies use a single-task approach, with one independent convolutional network per task. The segmentation tasks are essentially dealt with state-of-the-art like Mask-RCNN [He *et al.* 2017] and U-Net-like [Ronneberger *et al.* 2015a] architecture. The glaucoma classification task is instead mostly dealt with ResNet [He *et al.* 2016c], Inception [Szegedy *et al.* 2017] and DeepLab [Chen *et al.* 2017] networks, with eventual ensembling methods. Some contenders advantageously ensemble their network classification predictions with these of a linear classifier, trained over the vertical Cup-to-Disc Ratio (vCDR) obtained from the segmentation predictions.

In order to cope with the few provided labeled data, half of the contenders applied Transfer Learning from networks pre-trained on Imagenet [Deng *et al.* 2009]. A few other teams used existing related datasets for training. However, none of the teams in the challenge adopted a multi-task approach, although multiple related tasks were provided over a same input domain. One first explanation could be that the tasks are of different nature (classification and segmentation), and the state-of-the-art on these tasks is reached by different network architectures in the existing literature, making it inconvenient to share a same network for all tasks. A second explanation might come from task interference issues, leading to marginal gains or even degradation compared to single task learners.

One existing multi-task approach on this dataset has been exposed in [Zhao *et al.* 2019]. The authors propose a weakly supervised method, allowing them to predict glaucoma diagnosis, OD segmentation and a third evidence identification task with high accuracy, while only training with glaucoma diagnosis labels. Experimental results show comparable segmentation performance to a fully-supervised U-Net trained on the segmentation and evidence identification tasks. Although this thesis focuses on fully-supervised methods, this weakly supervised approach indicates well the close relationship between the classification and segmentation tasks, which shows that some benefits could definitely be expected from their joint learning.

## 5.3 Method

Instead of existing approaches [Hagiwara *et al.* 2018, Orlando *et al.* 2020] which generally train one deep CNN per task, here only one CNN is jointly trained on all the different tasks. This section first presents the whole adopted pipeline, with data pre-processing, network choice, and task prediction heads. It then introduces the

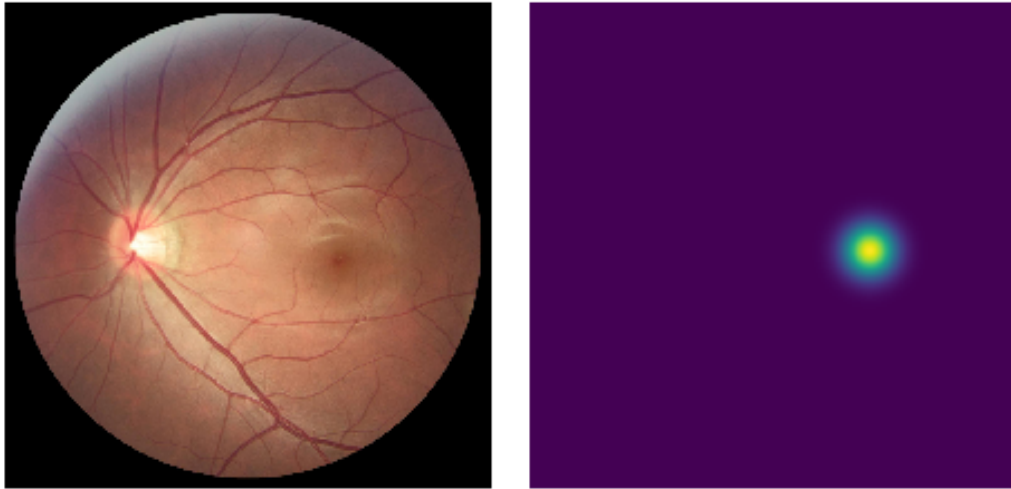


Figure 5.2: **Left:** An input sample (retinal fundus image). **Right:** Saliency map centered on fovea coordinates.

different loss functions and metrics used for the different tasks. It finally details how the different sharing strategies and optimization schemes presented in this thesis are applied to this pipeline.

### 5.3.1 Pipeline description

Figure 5.3 gives an overview of the whole pipeline, which is describe hereafter.

#### Tasks pre-processing

The dataset is composed of one binary classification task (glaucoma diagnosis), two semantic segmentation tasks (OD, OC), and on localization task (Fovea). Localization tasks are difficult tasks to process with convolutional networks, which do not use any notion of distance, but instead focus on receptive fields. Existing localization methods generally involve complex networks with anchor proposal mechanisms [He *et al.* 2017]. Instead, the fovea localization task is casted into a segmentation one: from the given coordinates of the fovea is created a saliency map as a bidirectional Gaussian centered in the fovea coordinates. An example is shown in figure 5.2 (right). Fovea locations are then computed as the center of mass of such saliency maps.

#### Network choice

Now disposing of one classification task and three segmentation tasks, a network able to jointly optimize these different tasks is needed. The chosen network is a U-Net [Ronneberger *et al.* 2015b], an encoder-decoder convolutional network, with a VGG-16 [Simonyan & Zisserman 2015] structure, and added skip connections between equivalent depths of encoder and decoder. These skip connections allow the

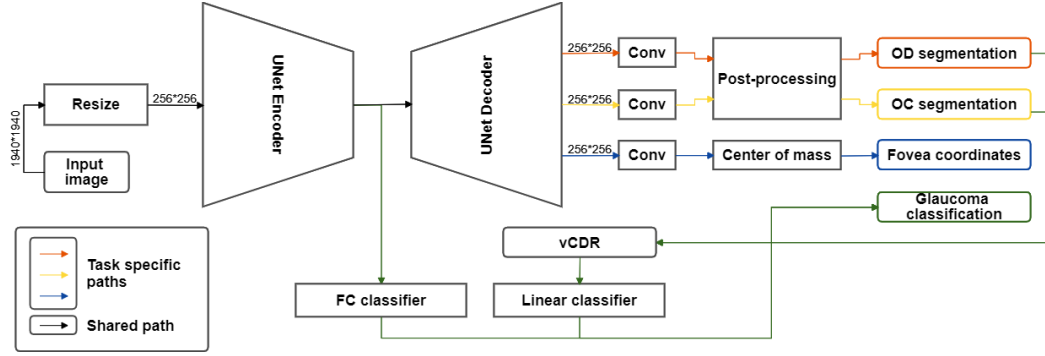


Figure 5.3: Method pipeline, used to deal with glaucoma diagnosis, OD/OC segmentation and fovea localization with a unique shared deep network.

decoder to recover fine-grained details through the multiple feature maps upscalings. This network is both well renowned and particularly efficient for segmentation tasks. Concerning the classification task, it can be processed by the VGG-like encoder part of the U-Net.

### Segmentation tasks predictions

The predictions for segmentation tasks are obtained with one task-respective convolutional layer after the shared decoder for each task. Similarly to existing works in [Orlando *et al.* 2020], the segmentations of OD and OC are then refined by only keeping the biggest connected instance in the prediction map, to remove eventual prediction noise around these elliptic regions. This is not applied to the fovea segmentation map, since it could shift its center of mass.

### Classification task predictions

The prediction for the classification task (glaucoma diagnosis) contains two steps:

- A first prediction is obtained from a fully connected layer, branched after the U-Net encoder.
- Similarly to some works in [Orlando *et al.* 2020], a second prediction is obtained from a linear classifier, taking as input the vertical Cup-to-Disc Ratio (vCDR) obtained from the OD and OC segmentation tasks. The vCDR is computed as:

$$vCDR = \frac{OC_{height}}{OD_{height}}$$

With  $OC_{height}$  and  $OD_{height}$  the heights of the optic cup and optic disc, computed on the segmentation maps.

The final classification score is obtained by averaging the predictions of the two classifiers.

### 5.3.2 Losses and metrics

Different loss functions and metrics are used for the different tasks to optimize.

#### OD and OC segmentation

The OD and OC segmentation tasks both use a binary cross-entropy loss (BCE), averaged over every pixel  $i$  of the segmentation maps:

$$\mathcal{L}_{BCE}(p, y) = -\frac{1}{N_{pix}} \sum_{i=1}^{N_{pix}} y_i \log(p_i) + (1 - y_i) \log(1 - p_i)$$

with  $p$ ,  $y$  and  $N_{pix}$  respectively the prediction, ground-truth and number of pixels. The metric used is the dice score (DSC), which computes the intersection over union between predicted and ground truth segmentation maps:

$$DSC(p, y) = \frac{2 \sum_i p_i y_i}{\sum_i p_i + \sum_i y_i}$$

#### Fovea localization

For the fovea localization task, the network is trained to fit the pre-processed saliency maps with a  $L1$ -loss, since values are not binarized on this task:

$$\mathcal{L}_{L1}(p, y) = \sum_i |y_i - p_i|$$

After the predicted fovea location is computed as the center of mass of the predicted saliency map, the metric used is the  $L2$ -distance.

#### Glaucoma classification

For the glaucoma classification task, I use a focal loss [Lin *et al.* 2017], to better handle the unbalance between positive and negative samples (only 10% of positives):

$$\mathcal{L}_{Focal}(p, y) = (1 - p_t)^\gamma \log(p_t)$$

with

$$p_t = \begin{cases} p & \text{if } y = 1 \\ (1 - p) & \text{otherwise} \end{cases}$$

Concretely, this loss multiplies the usual binary cross-entropy term with a classification uncertainty term  $(1 - p_t)$ , to give more importance to uncertain classifications, i.e. these of low populated classes.  $\gamma$  is a hyperparameter, set to 2 in the following. The metric used for this task is the Area Under Curve (AUC) score, in order to account for prediction confidence scores instead of just binary predictions.



### 5.3.3 Optimization

This chapter aims at comparing the performance obtained by the proposed multi-task pipeline with different combinations of sharing strategies (full sharing and Maximum Roaming partitioning) and optimization schemes (SUS, IUS and IO) discussed in Chapters 3 and 4. Different models represent different combinations of these strategies (six possible combinations in total), applied to the same pipeline.

For models using Maximum Roaming, the parameter partitioning is applied in every layer of the shared U-Net (encoder and decoder). The optimization schemes SUS, IUS and IO detailed in Chapter 4 are then applied over the loss functions of the four tasks. Note that for Maximum Roaming models, the SUS update scheme is not trivial, since these models require separate task-specific forward passes due to the partitioning. This combination MR-SUS is thus conducted by accumulating the losses of the different tasks forward passes, and back-propagating only when all task losses have been accumulated.

Along with the multi-task models, single-task versions (STL) of the same pipeline are also evaluated (adding four more models, one per task).

## 5.4 Experiments and Results

This section experiments the learning of the proposed multi-task pipeline on the four tasks of the REFUGE challenge. First are introduced some experimental details. Then are compared the obtained performance between all possible combinations of the two sharing patterns (full sharing and Maximum Roaming), and three optimization methods (SUS, IUS and IO) discussed in this thesis. The same pipeline is also run in single-task setting (independently for each task), in order to observe potential task interference phenomenon in the multi-task models. In a last part, the same study is conducted when combined with Transfer Learning, to evaluate if the proposed multi-task approach can produce benefits in this context.

### 5.4.1 Experimental details

For every pipeline setting, i.e. sharing pattern and optimization scheme combination, the learning is conducted with an Adam optimizer ( $\beta_1 = 0.9$ ,  $\beta_2 = 0.999$ ,  $\varepsilon = 10^{-8}$ ) and a batch size of 8. I use Maximum Roaming with a partition selectivity  $p = 0.8$  and an update interval  $\Delta$  of 1 epochs. I apply a dropout rate of 0.8 to the fully connected layer of the Glaucoma classification task. The network is trained for 300 epochs. After each epoch, the validation performances are ranked according to the challenge ranking procedure. All the code is made under Pytorch 1.2, and run on NVIDIA Titan XP graphic cards.

### 5.4.2 Experimental Results

For all models is first applied a grid search over the learning rate, to obtain a better insight on the different models behavior.

#### Grid-search reported on loss values

The results are reported in Figure 5.4 for fully shared models, and Figure 5.5 for Maximum Roaming models (MR). Specifically, the best validation losses achieved by the SUS, IUS and IO optimizations over a full training, w.r.t. different values of  $\eta$  are reported.

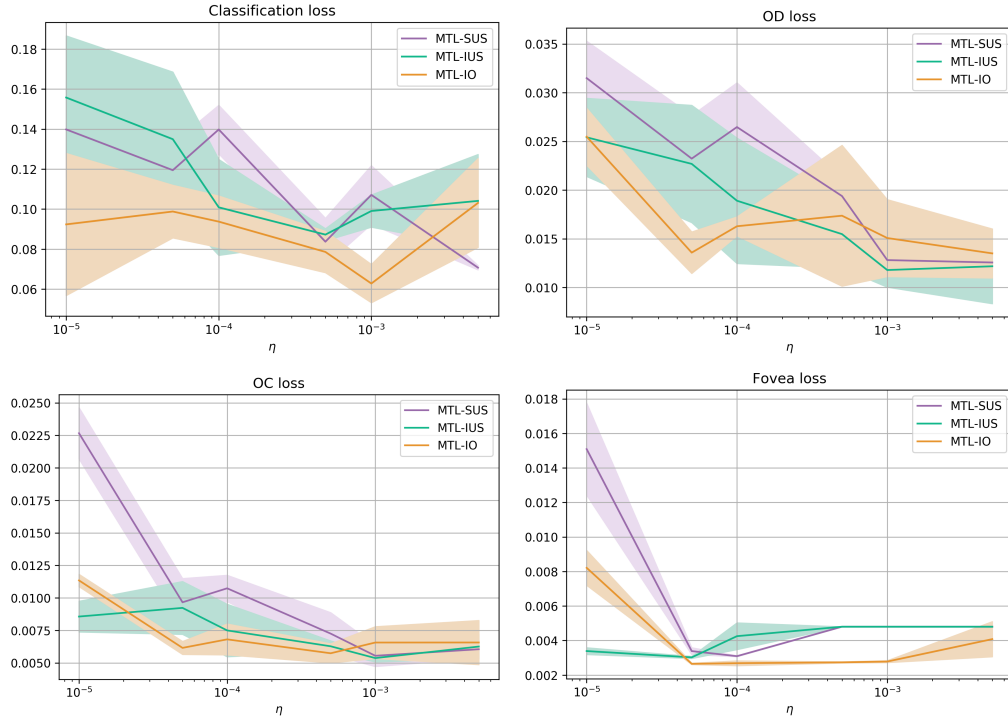


Figure 5.4: Reported losses of the learning rate grid-search over fully shared (MTL) models trained from scratch, with different optimization schemes (SUS, IUS and IO).

One observation from these two figures is the strong unbalance between the different loss magnitudes: some classification loss values are close to 100 times bigger than some fovea loss values, which means that the fovea localization task has chances to be dominated by the classification task, at the risk of degraded performances. Then, one important issue with these figures is that some learning dynamics are not accurately represented: in the case of the fovea localization task, some slight differences in the loss can lead to huge differences in the task metrics. In Figure 5.6 is represented the fovea localization validation metric through training of the MTL models with a learning rate  $\eta = 5e^{-3}$ . One can indeed observe that the SUS and IUS models completely overlook this task, while presenting a loss value close to that of

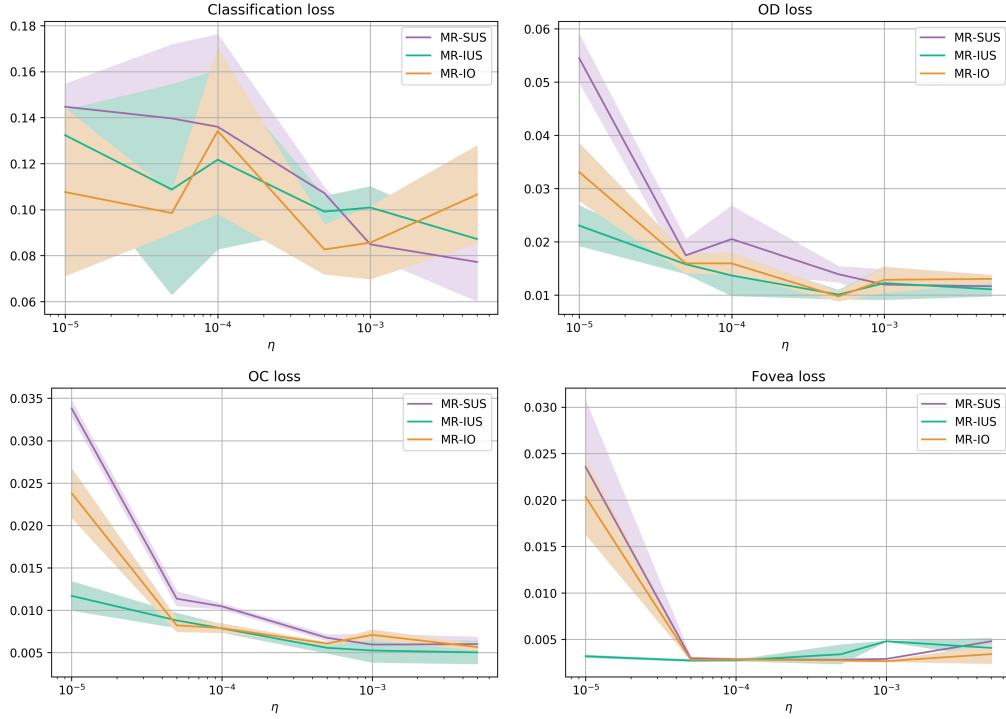


Figure 5.5: Reported losses of the learning rate grid-search over Maximum Roaming (MR) models trained from scratch, with different optimization schemes (SUS, IUS and IO).

the IO model. Instead, the different task metrics introduced in 5.3.2 might represent more accurately the true performances of the different models. Therefore, in the following the grid searches will be reported with respect to the task-specific metrics.

#### Grid-search reported on task metrics values

Figure 5.7 and 5.8 report the same grid search for fully shared and Maximum Roaming methods respectively, with task respective metrics reported instead of losses.

One can first see on these figures noticeable differences compared to the previous ones (reporting the tasks losses): in particular, the models failing on the fovea localization task are more visible, and the top performances achieved on the different curves are slightly shifted towards the lower learning rates for the classification task.

Both in the case of fully-shared and Maximum Roaming models, the IO optimization scheme consistently achieves better validation metrics over the different tasks and learning rates, and reaches the best performance over the grid search on every single task, often with a substantial margin. It is also interesting to notice that it reaches close-to-best performance on the fovea task on wider learning rate frames. Keeping in mind that this task has a loss of inferior magnitude, it suggests that the IO

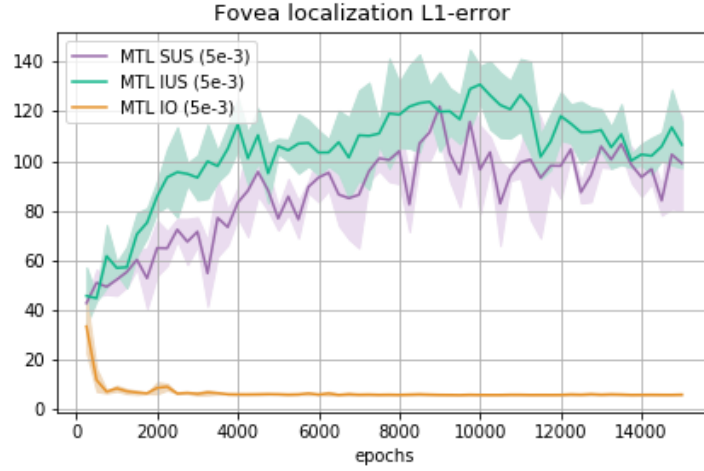


Figure 5.6: Fovea localization  $L1$ -error validation metric over training of the different MTL models, with a learning rate  $\eta = 5e^{-3}$ . While the loss values of SUS, IUS and IO models are pretty close for this learning rate, the difference is huge on the task metric.

optimization scheme better handles losses unbalance between tasks.

The contribution of Maximum Roaming can be observed by comparing the curves in Figures 5.7 and 5.8. With SUS and IUS optimization schemes, Maximum Roaming significantly improves the results achieved on segmentation tasks and fovea localization task. No notable differences are visible on the classification task. This suggests that the roaming of parameters among the different tasks during the optimizations helps to discover more favorable loss regions for tasks of lower loss magnitude, while it is more difficult for fully shared approaches to optimize them jointly with the other tasks. However the combination of MR and IO doesn't show the same improvements: the results are pretty similar to these obtained with the fully shared network and IO optimization scheme.

### Benchmark

Finally, Table 5.1 presents a benchmark, comparing the best models obtained from the previous grid search over fully shared and Maximum Roaming model, with the three different optimization schemes. In order to obtain a unified metric for comparing the different models, every obtained metric score is normalized between the best and worse performance achieved on the respective task among all models, which gives scores between 0 and 1 for every task. A multi-task score is then averaged from the four task scores. The best scoring models of each sharing strategy and optimization scheme at their best scoring iteration on the validation set are selected for comparison on the test set. Single task baselines (STL) for each task are also included in the benchmark.

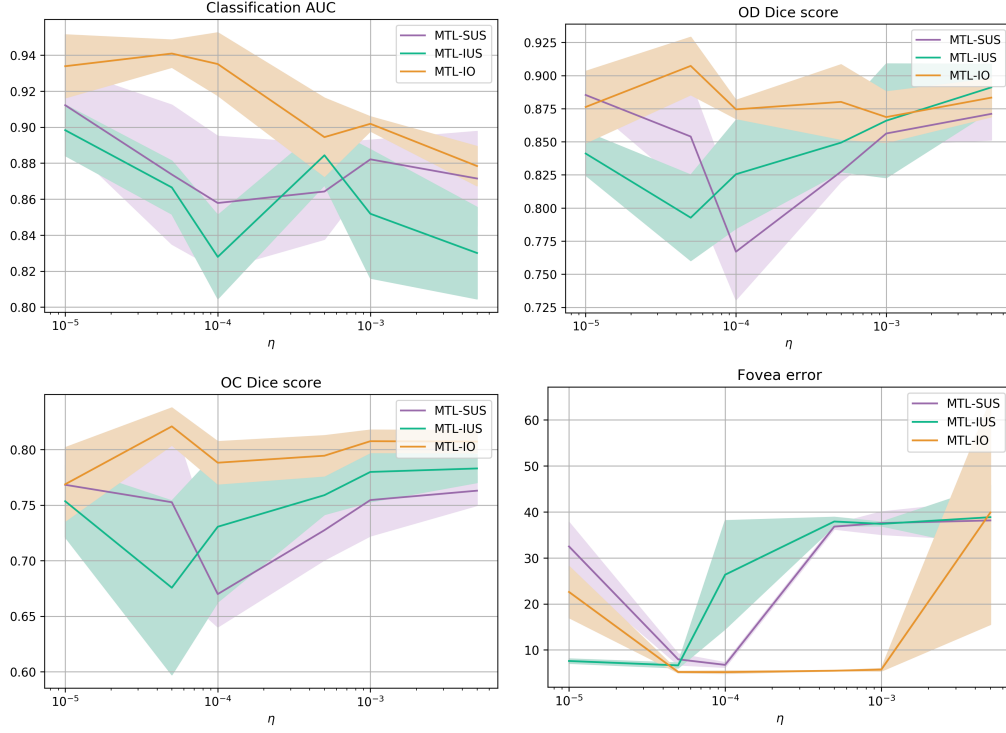


Figure 5.7: Reported metrics of the learning rate grid-search over fully shared (MTL) models trained from scratch, with different optimization schemes (SUS, IUS and IO).

The previous observations about the optimization schemes mostly hold here: the IO models reach the best scores on every single task among multi-task models (MTL and MR), by large margins. The IUS models instead seem like the worst ones, although the MTL-SUS model totally fails on the fovea localization task. Once again, Maximum Roaming seems to operate a better balancing between the four different task performances when combined with SUS and IUS optimization schemes, although the downside is lower performance on the classification task. The benefits of Maximum Roaming and the IO scheme do not cumulate, since the MTL-IO model performs slightly better than the MR-IO one, while cumulated effects can be observed on the IUS models. This may suggest that the MR-IO combination introduces too much noise in the optimization.

When comparing the performance of multi-task and single task models, it is interesting to notice that the fully shared network with standard SUS optimization scheme (MTL-SUS) performs worse than the single task baselines on every tasks, highlighting a task interference issue. Instead, when combined with the methods proposed in this thesis, the multi-task network is able to greatly reduce task interference, and often produce improvements compared to the single task baselines. In particular, the MTL-IO model reaches better scores on 3 of the 4 tasks, and closely approaches

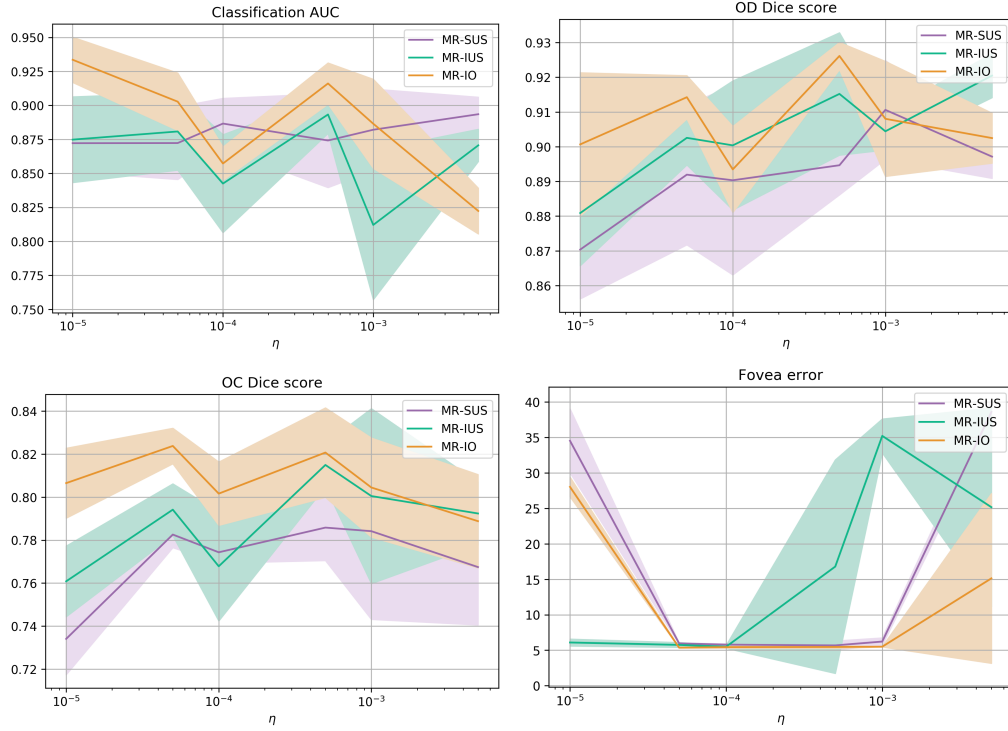


Figure 5.8: Reported metrics of the learning rate grid-search over Maximum Roaming (MR) models trained from scratch, with different optimization schemes (SUS, IUS and IO).

the STL score on the Fovea localization task.

### 5.4.3 Combination with Transfer Learning

Transfer Learning is a widely adopted method to bias a model with prior knowledge on an input domain and lead it to better generalization on new data. In practice, Imagenet [Deng *et al.* 2009] pre-trained models have proven to be profitable on a large majority of vision tasks. In medical imaging, although the input domain is different from the Imagenet domain (natural images), the benefits are still noticeable [Tajbakhsh *et al.* 2020], and particularly appreciated to compensate for the usual lack of training data. Its combination with Multi-Task Learning strategies studied here is thus relevant. This part is a similar study as the one presented above, this time built on top of a pre-trained model. As Imagenet only involves image classification, there exist no Imagenet pre-trained models for semantic segmentation. However, it is possible to use a pre-trained VGG-16 [Simonyan & Zisserman 2015] network for the encoding part of the U-Net in the pipeline, while the decoder is initialized from scratch.

As in the previous experiments, Figures 5.9 and 5.10 report the grid search operated

Models trained from scratch	AUC (↑)	Dice OD (↑)	Dice OC (↑)	Fov. Err. (↓)
STL (Classif.)	$90.09 \pm 2.70$	—	—	—
STL (OD)	—	$89.25 \pm 1.87$	—	—
STL (OC)	—	—	$80.61 \pm 1.36$	—
STL (Fovea)	—	—	—	<b><math>5.46 \pm 0.23</math></b>
MTL-SUS	$89.79 \pm 1.35$	$87.81 \pm 0.96$	$74.40 \pm 0.32$	$36.51 \pm 4.40$
MTL-IUS	$85.47 \pm 3.80$	$82.88 \pm 3.40$	$70.57 \pm 3.67$	$9.06 \pm 2.03$
MTL-IO	<b><math>92.91 \pm 0.69</math></b>	<b><math>92.07 \pm 1.50</math></b>	<b><math>81.37 \pm 0.20</math></b>	$5.60 \pm 0.24$
GradNorm	$86.31 \pm 1.35$	$80.32 \pm 1.10$	$68.93 \pm 4.78$	$28.48 \pm 0.65$
PCGrad	$85.51 \pm 1.71$	$85.23 \pm 1.52$	$71.08 \pm 7.40$	$34.45 \pm 0.33$

Table 5.1: Benchmark of the different combinations between different parameter sharing strategies (MTL and MR) and optimization schemes (SUS, IUS and IO), along with single task models (STL), trained from scratch.

on the task specific metrics, and Table 5.2 the benchmark comparing the best performing models.

Pre-trained models	AUC (↑)	Dice OD (↑)	Dice OC (↑)	Fov. Err. (↓)
STL (Classif.)	$94.30 \pm 1.68$	—	—	—
STL (OD)	—	<b><math>95.29 \pm 0.01</math></b>	—	—
STL (OC)	—	—	<b><math>85.86 \pm 0.21</math></b>	—
STL (Fovea)	—	—	—	$5.42 \pm 0.06$
MTL-SUS	$94.96 \pm 0.74$	$93.94 \pm 0.27$	$84.62 \pm 0.40$	$11.55 \pm 1.71$
MTL-IUS	$96.06 \pm 0.91$	$93.94 \pm 0.27$	$83.89 \pm 0.98$	$5.36 \pm 0.03$
MTL-IO	<b><math>96.15 \pm 0.14</math></b>	$94.24 \pm 0.38$	$83.95 \pm 0.90$	$5.22 \pm 0.18$
MR-SUS	$92.92 \pm 1.83$	$94.00 \pm 0.76$	$84.23 \pm 0.61$	$5.50 \pm 0.16$
MR-IUS	$94.57 \pm 0.53$	$94.10 \pm 0.10$	$84.31 \pm 0.31$	<b><math>5.20 \pm 0.24</math></b>
MR-IO	$94.82 \pm 0.84$	$94.63 \pm 0.24$	$83.86 \pm 0.22$	$5.51 \pm 0.09$
GradNorm	$88.06 \pm 3.17$	$81.37 \pm 6.91$	$67.69 \pm 9.75$	$32.38 \pm 1.11$
PCGrad	$95.24 \pm 0.67$	$94.41 \pm 0.38$	$84.56 \pm 0.40$	$5.34 \pm 0.13$

Table 5.2: Benchmark of the different combinations between different parameter sharing strategies (MTL and MR) and optimization schemes (SUS, IUS and IO), along with single task models (STL), used with a pre-trained network.

From a first look at the grid search figures, one can notice that Transfer Learning provides an overall performance gain for all models and tasks. The performance peaks tend to happen for smaller learning rates compared to the networks trained from scratch. The performance then quickly drops with the increase of  $\eta$ . This kind of behavior is generally expected when using Transfer Learning: the

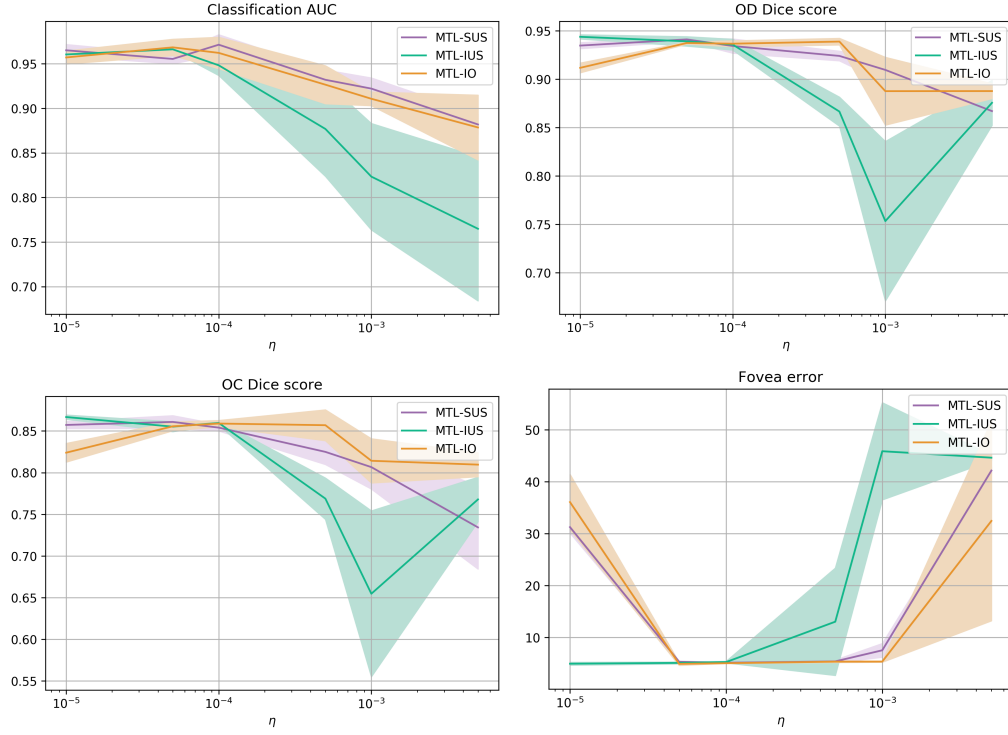


Figure 5.9: Reported metrics of the learning rate grid-search over pretrained fully shared (MTL) models, with different optimization schemes (SUS, IUS and IO).

pre-trained weights provide robust representations on the source input domain trained on millions of images, which needs only small adaptations in order to adapt to the slight domain shift. A too large learning rate might instead break some of these representations, and thus waste the provided inductive bias [Yosinski *et al.* 2014, Tamaazousti *et al.* 2017]. Pushing further this reasoning, the fact that this phenomenon is less pronounced on the segmentation tasks might come from that they use a decoder initialized from scratch, therefore reducing the impact of the pre-trained encoder.

Then, the benefits obtained by Maximum Roaming and the optimization schemes IUS and IO are similar to these obtained without Transfer Learning: both the optimization schemes consistently increase the performance on the different tasks, while Maximum Roaming models seem to give more attention to tasks of lower loss magnitude, at the cost of a performance drop on the classification task. However, the gains produced by these methods here are of smaller magnitude compared to these obtained on models trained from scratch, which could suggest that the pre-training places the network in smoother regions of the loss landscape, in which different optimization and sharing strategies make less differences. Similar conclusion can be drawn from comparison with the STL models, which are also slighter: Transfer Learning obviously reduces the impact of Multi-Task Learning.



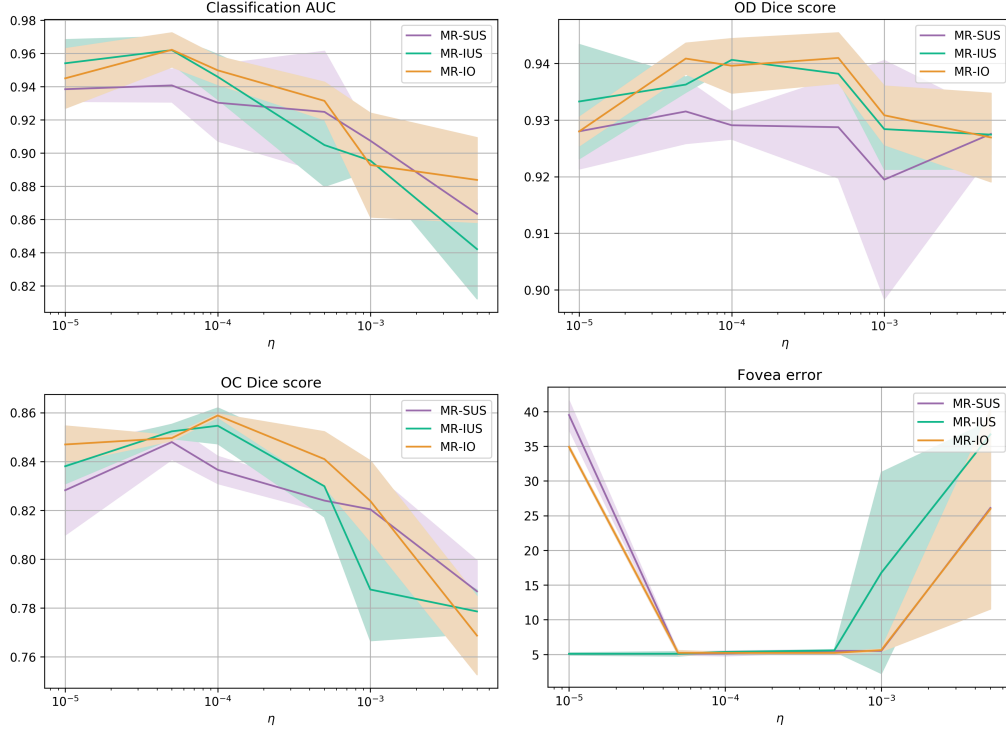


Figure 5.10: Reported metrics of the learning rate grid-search over pretrained Maximum Roaming (MR) models, with different optimization schemes (SUS, IUS and IO).

Finally, the fact that Maximum Roaming behaves in a similar way when combined with Transfer Learning is particularly interesting: due to the parameter partitioning operated in MR models, one could expect to break some co-adaptations in the pre-trained layers, and thus reduce the impact of Transfer Learning, which doesn't seem to be the case here.

## 5.5 Discussion

This chapter proposes the application of the multi-task methods discussed in this thesis on a real world medical imaging challenge centered on glaucoma diagnosis, for which methods to bias the learning process like Transfer Learning and Multi-Task Learning are of particular relevance to compensate for the lack of annotated data. The proposed pipeline aims at maximising the inductive bias by using a unique CNN to learn all tasks jointly, and applies different combinations of sharing strategies and optimization schemes to compare their respective impacts on generalization and task interference. These same experiments were then conducted when using a pre-trained network, i.e. Transfer Learning, in order to evaluate how hold these methods benefit when combined with respect to this other learning paradigm.

The experimental results on networks trained from scratch mostly confirm the findings of Chapters 3 and 4, and provide interesting information about their combination:

- Maximum Roaming proceeds of an overall improvement of the multiple tasks: in a setting where one task loss dominates the others, the dominated tasks see their performance significantly increased, at the cost of a moderated performance loss on the dominating task.
- The optimization scheme using independent optimizers (IO) for each task greatly improves the performance on every task, while results are more comparable between the shared optimization scheme and independent one with same optimizer (SUS and IO).
- The combination of Maximum Roaming. and IO scheme do not improve the results of IO used alone, while it is the case with IUS.
- The proposed methods greatly reduce task interference, and consistently out-class single-task baselines.

This is to add to the fact the presented multi-task pipeline uses a unique convolutional network for all tasks, which is more lightweight compared to single-task learners. Therefore, the proposed pipeline combined with the methods presented in this thesis produces significant benefits both in terms of performance and computational efficiency.

These observation hold in smaller proportions when these methods are combined with Transfer Learning, which suggests that these strategies can be efficiently applied in real world contexts to create better generalization performance on Multi-Task problems, in isolation, or eventually combined with Transfer Learning when possible. However more investigation should be conducted under different setting and problems to obtain more insight about the combination of MR and IO.



# Conclusion

---

## Contents

<b>6.1</b>	<b>Summary of contributions . . . . .</b>	<b>79</b>
<b>6.2</b>	<b>Open perspectives . . . . .</b>	<b>81</b>

---

## 6.1 Summary of contributions

Over the last years, research in computer vision tried to make the most benefit out of limited existing annotated data to introduce inductive bias into their learning processes. Among different existing solutions is Multi-Task Learning, which consists in jointly optimizing a unique model with respect to multiple tasks. The presence of multiple tasks issued from a same domain provides more information to the network about the input domain, which should improve its generalization on new data. However, current research encounters major difficulties in the optimization of deep multi-task networks, which in practice do not consistently reach better generalization performance compared to equivalent single-task learners, and can instead degrade it. This is denoted as *task interference*. This thesis proposed two methods to improve the optimization of deep multi-task networks and mitigate task interference. The relevance and feasibility of the proposed methods was demonstrated through its application to a real world concrete problem involving the detection of glaucoma from fundus images.

### Maximum Roaming (Chapter 3)

Parameter partitioning methods explicitly propose to combat task interference by relaxing the standard hard parameter sharing strategy: each task is attributed a specific use of the parameters in each layer. This procedure experimentally shows to efficiently reduce task interference. However existing partitioning methods might block interactions between some tasks and parameters and therefore reduce the latent inductive bias. This thesis proposed Maximum Roaming, a dynamic parameter partitioning method inspired by dropout. The discrete parameter partitioning is regularly updated through learning with respect to a finite update plan, which imposes every parameter to be trained by every single task for a certain duration. In the meantime, the averaged overlap between the different task partitions remains constant, in order to fully benefit of the interference reduction

provided by partitioning methods. The method is model-agnostic, only involves negligible supplementary non-trainable binary parameters. Experimental results show consistently improved performance across multiple datasets and different convolutional networks over existing state of the art baselines.

### **Alternated and independent optimization of tasks-specific objectives (Chapter 4)**

Partitioning methods, due to the task-specified parameters usage, compute separate forward and backward passes for each task, sequentially. This optimization scheme is different from most other approaches, back-propagating at once all the different losses aggregated in a multi-task loss. However this difference is never studied in isolation in the literature. This thesis provided a convergence analysis of this optimization scheme, named IUS (Independent Update Steps) here. It was then compared with the standard optimization of the aggregated loss. Suspecting that giving more independence to the tasks could introduce more stochasticity into the optimization and reveal beneficial, a new optimization scheme was then proposed, defining one optimizer per task (Independent Optimizers, IO), to avoid the mixing of different tasks gradients into the momentum mechanisms of state-of-the-art optimizers. Experimental results report that these two optimization schemes, with a clear advantage to the IO one, consistently achieve better performance than the aggregated loss optimization scheme, and compare favorably to other state-of-the-art baselines. They also report a larger traveled distance into the loss landscape, reaching more remote solutions, with more oscillations on their path, suggesting that they indeed bring more stochasticity into the optimization process. However, in the worst case, both the IUS and IO schemes scale linearly to the number of tasks, in terms of optimization time. Experiments are therefore conducted on their combination with a simple task grouping strategy, which allows to freely balance the optimization schemes benefits with the optimization time increase. Experimental results over three datasets show both MTL-IUS and MTL-IO achieve an overall better generalization performance w.r.t standard aggregated objective optimization (MTL-SUS) and state-of-the-art MTL baselines using an aggregated objective function. In particular, MTL-IO shows important improvements in settings where tasks are of a very different nature. These results also confirm that the proposed random grouping strategy applied to MTL-IUS and MTL-IO is beneficial both in terms of performance and efficiency, when dealing with a great number of tasks. Finally, the results showed that MTL-IUS and IO allow parameters updates to travel a longer distance, and ensure a more thorough exploration of the shared parameter space.

### **Application on a medical imaging challenge (Chapter 5)**

Concerned about how well the proposed methods could generalize over non-academic contexts, this thesis evaluated them on a medical imaging challenge, centered on glaucoma diagnosing from fundus imaging, and providing annotations for multiple tasks. A Multi-Task Learning approach was adopted to create the most possible

inductive bias from the few data at disposal, and optimize with different possible combinations of sharing strategies (i.e. fully shared and Maximum Roaming models) and optimization schemes (SUS, IUS and IO). The comparisons showed that the contributions found in the previous chapters are also observable in this context, with improved multi-task performance. While the optimization schemes IUS and IO consistently improve each task performance, Maximum Roaming benefits are more focused on tasks presenting lower loss magnitude. Finally, this thesis studied how Multi-Task Learning could be combined with other strategies, such as Transfer Learning.

## 6.2 Open perspectives

The most direct perspectives opened by this thesis are the possible improvements of the two introduced methods, Maximum Roaming, and the separate and independent optimization of the tasks, MTL-IO.

### **Towards learnable roaming parameter partitioning**

In opposition to other partitioning methods which try to find an optimal partitioning, regardless of the potentially lost latent inductive bias, Maximum Roaming instead only focuses on the roaming of parameters among tasks to create more regularization, and proceeds random updates regardless of any parameter-task affinity. Further improvements could unify both these different approaches, in order to fully exploit the latent inductive bias, while maximising the efficiency of the partitions with respect to their tasks.

A first solution would be to apply the two approaches sequentially: one first training with Maximum Roaming to introduce robust knowledge into the parameters, followed by a fine-tuning of the partitions to refine better task-parameters combinations. A second possibility would be to find a differentiable representation of the roaming of parameters and introduce it in the loss function, so that parameters optimization, partitions optimization and roaming are all conducted jointly. Such differentiable approach would also avoid the hyper-parameter tuning in Maximum Roaming.

### **Re-thinking optimization schemes in existing MTL works**

Chapters 4 and 5 showed that alternated and independent optimization of tasks (MTL-IO) consistently achieved better generalization performance than the standard optimization of an aggregated loss. However, its main downside remains the increased optimization time due to task-specific update steps. Experimental results thus showed that a naïve random task grouping strategy is enough to reduce at will this optimization time, at the cost of reduced performance gains. One straightforward evolution would be to replace the random task grouping with sophisticated task grouping methods, as some of these pre-

sented in Chapter 2, in order to limit the risks of negative transfer among task groups.

Then, MTL-IO is proposed as an alternative way of jointly optimizing a set of parameters with respect to multiple tasks. As shown in Chapter 5, it is not specifically intended to be used in isolation, but can instead be combined with most existing multi-task works, with minimal effort. It is therefore encouraged to further research on how to apply this simple optimization scheme to other existing multi-task methods, in order to eventually combine their benefits.

# Appendices





# Semantic and Visual Similarities for Efficient Knowledge Transfer in CNN Training

---

## Contents

---

<b>A.1 Introduction</b>	<b>85</b>
<b>A.2 Related Works</b>	<b>87</b>
A.2.1 Datasets and architectures	87
A.2.2 Transfer Learning process	87
A.2.3 Semantic similarity between textual content	88
<b>A.3 Similarity-based knowledge transfer</b>	<b>89</b>
A.3.1 Similarity measures	89
A.3.2 Initialization	90
<b>A.4 Experiments and Results</b>	<b>90</b>
A.4.1 Implementation Details	90
A.4.2 Dataset	91
A.4.3 Similarities and Initialization	91
A.4.4 Neighboring optimization	92
A.4.5 Data reduction study	94
<b>A.5 Conclusion and Perspectives</b>	<b>95</b>

---

The content of this chapter is based on: "Semantic and Visual Similarities for Efficient Knowledge Transfer in CNN Training" [Pascal *et al.* 2019], which was published as a conference paper in CBMI 2019.

## A.1 Introduction

With the emergence of large, public and thoroughly annotated datasets [Russakovsky *et al.* 2015], along with the ever increasing computing capabilities of GPUs, Deep Neural Networks, and especially CNNs, have rapidly revolutionized many computer vision tasks. Such quantities of data allow to learn visual feature extractors whose relevance and discrimination power surpasses the best hand crafted ones [Krizhevsky *et al.* 2012], regardless of the problem complexity or the model

size. However, the time and associated cost for creating such new huge datasets, and to make new models converge over these are still a huge bottleneck in real-world use cases, so that someone with limited resources cannot reasonably compete with companies running each of their models during weeks over hundreds of GPUs (or TPUs) and gigantic datasets.

Transfer learning is a recent and still evolving approach to address this issue. It consists in reusing a model developed for a task as a starting point for another related task. This is based on the assumption that two related tasks require some common knowledge, so that some of the knowledge associated to a task could benefit another similar one. In deep learning, this is performed by using some of the model's weights as an initialization for the training over the new task, while the usual practice is to initialize them randomly ([Glorot & Bengio 2010]). In computer vision tasks, well trained CNN low level and mid-level layers generally detect basic shapes and textures, no matter the specificity of the task. They consequently transfer well between different computer vision problems, as shown in numerous publications ([Hinton & Salakhutdinov 2006, Yosinski *et al.* 2014, Chu *et al.* 2016, Azizpour *et al.* 2016, Zamir *et al.* 2018, Tamaazousti *et al.* 2017, Wang *et al.* 2017]).

Transfer learning can then be distinguished in two types of applications : domain adaptation, aiming to adapt a pre-trained network to a new task, out of this work's scope, and fine-tuning, which consists in adapting the network to new target data, for the same task. In the latter case, one can extend the transfer to the whole set of weights concerning the features extraction, and just replace the output fully connected layer with one shaped for his target classes. Almost all the knowledge required to perform the task is already present in the network, and can be aligned with the target data by a small training procedure (not necessary on the entire set of weights), lighter than the one required to train a model "from scratch". Note that fine-tuning a pre-trained CNN often leads to better performances, and requires fewer data than a network trained from scratch, as most of the knowledge required for the task is already present in it ([Yosinski *et al.* 2014, Chu *et al.* 2016]).

Fine tuning has become common practice, allowing faster trainings on consequently smaller datasets, and giving the opportunity for researchers and companies to develop their own systems. However, there may still be a lack of efficiency in training a new fully connected layer from scratch, and transfer learning can once again fulfill it. To further improve the transfer, this chapter thus proposes to reuse some weights of the last fully connected layer of the original model, based on similarity between source and target classes.

The contribution of this work is fivefold: First is showed that there is some important knowledge within the last layer of pre-trained DNN models which when identified and used properly can be somewhat transferred to the new model, to

speed up training and benefit model accuracy for fine tuning. Second, a novel method is proposed to reuse that knowledge in combining multiple relevant source classes. Third, a study is conducted over one visual and two semantic similarity measures to select these relevant source classes. Fourth, an original analysis enables to separate cases between three possible types of knowledge transfer, to attest that the proposed method performs well on each of them, and optimize the proposed method. Fifth, the proposed method is monitored while decreasing the amount of training data, to validate the consistency of the results.

This chapter is organized as follows. It first reviews some related works, then presents the contribution, before discussing the experimental results. A final discussion is provided on this work and future developments.

## A.2 Related Works

### A.2.1 Datasets and architectures

The most common source dataset to apply transfer learning for computer vision tasks is ImageNet [Russakovsky *et al.* 2015], since it presents 1000 classes, shared between various semantic fields (animals, flora life, vehicles, tools, etc...). It is thus very likely to benefit the training of almost any kind of target data, and its efficiency is demonstrated in [Huh *et al.* 2016].

As for the choice of the network to use, many state of the art results in image classification tasks (including the ImageNet dataset) have been achieved by (or built on) the ResNet architecture [He *et al.* 2016a, He *et al.* 2016b], and Inception [Szegedy *et al.* 2015, Szegedy *et al.* 2017] structures (or combinations of them). Their efficiency and simplicity often places them as the best choice for transfer learning, be it for other classification tasks, or using it as a backbone for other tasks (object detection and segmentation, for example).

### A.2.2 Transfer Learning process

Reusing some pre-trained weights for a new task has been pioneered by [Hinton & Salakhutdinov 2006] and [Yosinski *et al.* 2014], showing that the new task can greatly benefit it, not only in terms of training speed, but also of global performance.

However, the way to optimize a transfer learning process is still unclear. It is known that the deeper the layer in a network, the more specialized are its weights to the task they are trained on [Krizhevsky *et al.* 2012, Yosinski *et al.* 2014]. Concretely, if the first few layers of a ResNet (detecting simple visual patterns like geometrical shapes) can benefit any computer vision task, it is still unclear how deep the weights

can efficiently be reused. [Yosinski *et al.* 2014] experiments transfers of different depths, and highlights that the transfer of specialized layers can hurt performance on the target task, depending on their depth.

In [Chu *et al.* 2016] is given a study taking into account the amount of data available. They show that if transferring weights has only a moderate impact on performance in a context with a lot of data, it becomes crucial for long-term performance as the data decreases. For two sufficiently close tasks (source and target), they generally advise to transfer all the layers except the classification one before a global fine tuning. This advice seems quite reasonable in the case investigated here, since only the images contained in the dataset and their classes change, while the classification objective remains the same.

However, with enough populated classes, [Yosinski *et al.* 2014] shows that training only the randomly initialized part of the transferred CNN can break fragile co-adaptations at the boundary. Fine tuning equally the whole network gives better results, allowing to readjust those co-adaptations. [Tamaazousti *et al.* 2017] proposes a finer process that consists in training the whole network in one go with a lower learning rate applied to the transferred part. This focuses the training on the new part, while allowing co-adaptation between the two parts.

As shown in [Wang *et al.* 2017], transfer learning can also be improved by deepening and/or widening the original network, giving more rooms to small adjustments, under the condition of correctly managing the simultaneous training of both transferred and newly created cells.

A systematic process in all these works is to discard the classifying layer and to train a new one, adapted to the target classes, from scratch. This chapter argues and shows that, when the source and target are similar to a certain extent, the knowledge contained in the pre-trained classifier layer can be efficiently reused for learning a new model.

### A.2.3 Semantic similarity between textual content

One traditional way to get a similarity measure between two concepts is to use the WordNet graph [Beckwith *et al.* 1991]: WordNet is an english lexical database of nouns, verbs, adjectives, and adverbs grouped under lexicalized concepts (named synsets), interlinked by different types of semantic relations. There are five main semantic similarity measures defined for WordNet in the literature : Jiang & Conrath [Jiang & Conrath 1997], Leacock & Chodorow [Leacock & Chodorow 1998], Lin [Lin *et al.* 1998], Resnik [Resnik 1995], and Wu & Palmer [Wu & Palmer 1994]. Each of them evaluates the semantic distance between two synsets. [Capelle *et al.* 2012] evaluated the Wu & Palmer one, making

use of the path length between the synsets organized in a 'is-a' hierarchy and the depth of their most specific ancestor node, as the best one for semantic similarities.

More recently, [Mikolov *et al.* 2013] designed an approach using neural networks to project words into feature vectors named Word2Vec representations, to represent efficiently textual content. In this feature space, distances between words are shown to be quite accurate to attest and quantify some semantic relationships [Mikolov *et al.* 2013, Wang 2014, Handler 2014].

### A.3 Similarity-based knowledge transfer

A standard, basic transfer learning process to train an image classifier for some target classes is to reuse the convolution weights of a network already trained on a similar task on source classes. A fully connected layer fitting the target task is then randomly initialized on top of the network, which is fine tuned following a strategy adapted to the specificities of the problem (amount of available data, possible computational power restrictions, etc...).

The fully connected layer of the pre-trained network represents the knowledge of the task it is devised for. In cases where the original classification problem and the destination one are close, there might be a gain in transferring some of the knowledge of the original network head (last layer) to the target one.

The hypothesis is made here that re-adjusting this available knowledge to fit the target classes could be more efficient than creating it from scratch, in the usual way. Assume there are  $M$  source classes and  $N$  target ones at disposal, each target class (represented by its fully connected weights) could be initialized with a combination of a relevant subset of those  $M$  source classes, instead of randomly. The relevance of such a subset of classes is defined by using alternative similarity measures.

#### A.3.1 Similarity measures

Three of them are proposed. The first is a visual one, directly based on the image content. The two others are label-based semantic similarities :

**Inference similarity.** The images of the target classes are input to the plain source network. The similarities between source and target classes are computed as F-score measures for each "source network output/target class" couples. In a more practical way, let  $o_j$  be the  $j^{th}$  output of the source network and  $c_i$  the  $i^{th}$  target class with  $j \in \{1, \dots, M\}$  and  $i \in \{1, \dots, N\}$ ,  $sim(i, j)$  is computed as the F-score for  $o_j$  discriminating  $c_i$ . This similarity measure aims at leveraging relations based

more on pure visual content than semantics.

**WordNet similarity**, using the Wu & Palmer ([Wu & Palmer 1994]) measure, as advised in [Capelle *et al.* 2012].

**Word2Vec similarity**. Using some pre-trained Word2Vec embeddings, the standard cosine similarity is computed between the Word2Vec embeddings of the source and target class names.

In the following section, these three initialization techniques are compared to the classic one (i.e. using random initialization of the neural network weights).

### A.3.2 Initialization

The affinity values are considered as the coefficients of a neighboring structure, allowing us to approach the target class as a combination of some source neighbors. For each target class are thus computed the weights of its classifier as a linear combination of its  $K$  closest source neighbors with respect to the similarity measure, taking as coefficients these affinity values (normalized, for them to sum to one over  $K$ ).

$$W'_i = \sum_{j=1}^K \left( \frac{sim(i, j)}{\sum_{j=1}^K sim(i, j)} \right) W_j$$

In this way, each of the  $K$  source classes neighbors contributes to the construction of the target class initialization, in proportion to its normalized similarity. The setting of  $K$  with respect to the target classes is studied in the experimental part.

## A.4 Experiments and Results

### A.4.1 Implementation Details

In the following, a ResNet-101 pre-trained on Imagenet is used to combine architecture simplicity and high performances, and replace the last fully connected layer to fit the target classes. Weights are trained from the fourth block (included) to the end, while the rest remains frozen. One could use a finer transfer strategy to optimize the results obtained [Yosinski *et al.* 2014, Chu *et al.* 2016, Wang *et al.* 2016, Tamaazousti *et al.* 2017]. Input images' smallest sides are resized to 256 (preserving aspect ratio), then cropped (randomly for training, center crop for testing) to output  $224 \times 224$  images.

The Adam optimizer is used with a learning rate of  $10^{-3}$ ,  $\beta_1 = 0.9$ ,  $\beta_2 = 0.999$  and  $\varepsilon = 10^{-8}$ , with a batch size of 64. A dropout of 0.75 is applied on the last fully connected layer to prevent overfitting.

#### A.4.2 Dataset

For this experiment, the 1000 classes of the ILSVRC challenge [Russakovsky *et al.* 2015] which the ResNet-101 has been trained to classify are used as source classes. For the target classes are selected 90 ImageNet synsets that are not part of those former 1000. They can be separated in three types :

**Included classes.** The target synset is a child of a source synset, thus representing a more restrictive class than the one in the original problem.

**Inclusive classes.** The target class is an ancestor of some source synset(s), thus representing a more general class.

**Disjoint classes.** Neither child nor ancestor of any already source synset.

For these 90 target classes, synsets containing at least 1000 images are selected, and equally distributed into the three types of target classes (30 classes each). Within each target class, 100 images are picked for testing and 900 for training, producing balanced training and testing sets. Depending on the experiments, only a certain portion of this training set will be used for training. The list of synsets used for this experiment along with the selected images is available on a GitHub repository <sup>1</sup>.

#### A.4.3 Similarities and Initialization

The inference similarities are computed as explained earlier, with a pass of the training set through the pre-trained network (with the 1000 class pre-trained classifier).

For the WordNet similarities, the WordNet module of the NLTK Python library is used to obtain a similarity measure based on the shortest path that connects the labels (or synsets) in the "is-a" (hypernym/hyponym) taxonomy.

To compute the Word2Vec embedding of a given label, the embeddings <sup>2</sup> of all the words composing it are averaged, since labels are not always denoted by a single word, but often by an expression.

<sup>1</sup><https://github.com/lucaspascal/semantic-and-visual-similarities-for-efficient-knowledge-transfer-in-CNN-training>.

<sup>2</sup>as trained on Flickr, and publicly available at <https://github.com/li-xirong/hierse/blob/master/README.md>.



Each target class is then initialized with the weights of its selected source(s), depending on the chosen similarity measure and the number of source neighbors. A standard Xavier initialization [Glorot & Bengio 2010] is used for the random initialization baseline.

#### A.4.4 Neighboring optimization

The first experiment shows the global behavior of the proposed initialization method with a single source class neighbor as initialization for each of the target class, in terms of F-score measure, averaged over the 90 target classes. For this experiment, each of the 90 target classes are populated with 500 training images (limit above which the results did not change significantly). Fig.A.1 shows the evolution through training in terms of F-score averaged over all the 90 target classes, with the four initialization strategies (i.e Random, Inference, WordNet and Word2Vec).

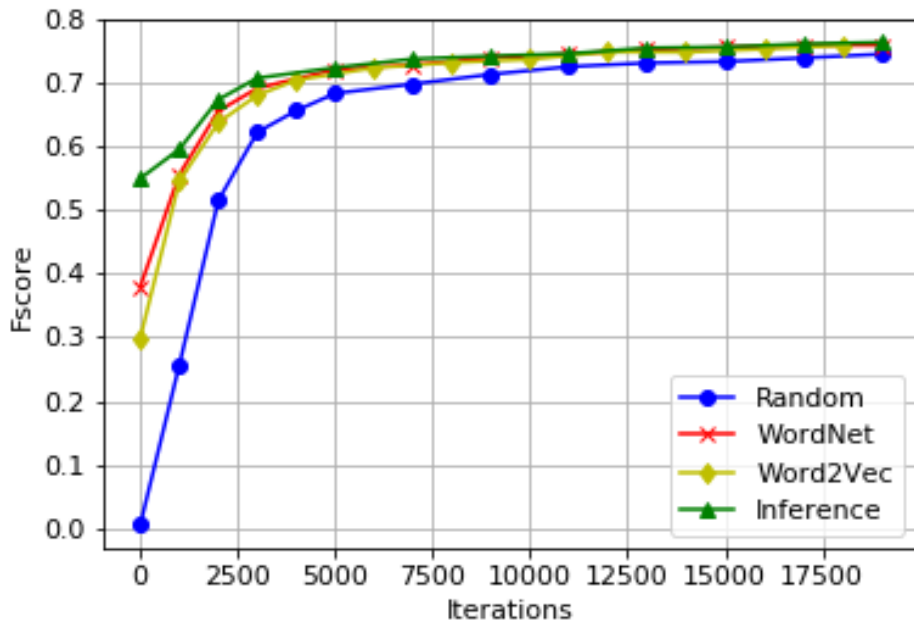


Figure A.1: averaged per-class Fscore

One can observe that each of the three initialization strategies performs better than the random baseline: the convergence is accelerated, and the models produce interesting results even without training (iteration 0): from 40% to more than 70% of the F-score achieved at convergence, depending on the model. The initialization by inference similarity is performing best, as one could have expected since the similarities in this case have directly been evaluated with respect to the task’s performance metric. The four models tend to converge to the same value, provided





with enough data to fill the gap.

In **Table A.1** is provided an example of source/target class correspondence given by each similarity measure, for the "*Anchor, ground tackle*" target class. The results of these transfers are shown in **Fig. A.2**. In this case, the Word2Vec method has been mistaken by the "*tackle*" term, and chose the football accessory as a source class (worst performing). WordNet found a logical source class ("*Hook, claw*"), according to the synsets semantic, and the inference similarity selected "*Sundial*", which has no obvious semantic link with an anchor, but presents some very similar visual patterns, as shown on the images (performing best).

From this example, along with the global results, can be drawn the conclusion that label-based semantic similarities are more likely to select wrong matchings for visual classification, while the inference similarity is able to bring out better ones, out of any semantic consideration

Another interesting fact here is that the Word2Vec similarity is still learning faster than the random initialization. This observation, verified over multiple other examples, suggests that any pre-trained classifier is always a better initialization than a random one for fine-tuning in image classification.

Table A.1: Classes correspondances

Target Class	Inference Affinity	WordNet Affinity	Word2Vec Affinity
Anchor, ground tackle	Sundial	Hook, claw	Football helmet
			

The study is then extended to the use of multiple source classes neighbors to compute the different initializations: **Fig. A.3** shows the F-scores of the models directly after initialization (without training), with respect to the number of source class neighbors selected, for each type of target class (i.e disjoint, included and inclusive).

The initialization by inference similarity benefits the most from extending the

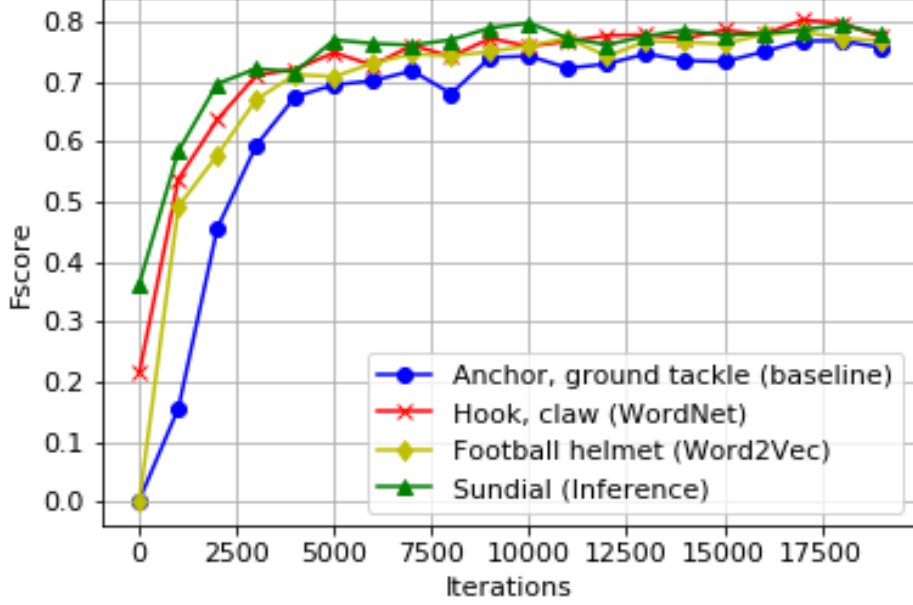


Figure A.2: Evolution of the models on the target class *Anchor, ground tackle*, for the different source classes determined by the similarity measures.

number of source class neighbors: for any type of initialization, the built classifiers smoothly gain in performances by adding neighbors. Beyond the selection of one best source class, this confirms the superiority of the visual similarity over the semantic ones to estimate the relevance of any source class for a transfer. For the WordNet and Word2Vec cases, there is also a significant gain, even if the evolution over the number of neighbors is more chaotic, and it appears to be a good way to compensate bad matchings (like the Word2Vec case in [Table A.1](#)).

#### A.4.5 Data reduction study

In this experiment is studied how well this process generalizes while decreasing the amount of training data. For each of the initialization methods, a new model is initialized, taking for each target class the optimal number of neighbors source classes depending on its type (disjoint, included or inclusive). These optimal numbers are taken from [Fig.A.3](#). [Table A.2](#) shows the scores of these models compared to the random baseline for 100, 50, 25, 10, 5, 2 and 1 training images per class. For each model, the initial performance after initialization (without training) and the best registered performance until convergence are reported.

The source classes selection by inference similarity varies with the number of



Figure A.3: Immediate inference results for each type of classes and initialization, with respect to the number of source classes selected to compute the initialization.

Table A.2: Evolution of the four methods through data reduction. Performances after initialization (left columns) and best registered performances (right columns) are reported, with respect to the number of training samples for each class.

Images per class	Random		Visual similarity		WordNet semantic similarity		Word2Vec semantic similarity	
	First	Best	First	Best	First	Best	First	Best
100	0.00	0.72	<b>0.59</b>	<b>0.73</b>	0.39	0.72	0.35	0.72
50	0.00	0.68	<b>0.58</b>	<b>0.69</b>	0.39	0.68	0.35	0.68
25	0.00	0.62	<b>0.58</b>	<b>0.64</b>	0.39	0.63	0.35	0.63
10	0.00	0.53	<b>0.54</b>	<b>0.54</b>	0.39	0.54	0.35	0.53
5	0.00	0.41	<b>0.50</b>	<b>0.50</b>	0.39	0.43	0.35	0.45
2	0.00	0.26	<b>0.44</b>	<b>0.44</b>	0.39	0.39	0.35	0.35
1	0.00	0.16	<b>0.40</b>	<b>0.40</b>	0.39	0.39	0.35	0.35

training images (unlike the two others), since it is computed with those images. Its performance at initialization thus decreases with the amount of training data. However, it still always achieves better performances, which puts aside the idea of combining both types of similarities [Safadi *et al.* 2014]. Under 5 training images per class, a consequent performance gap remains between the baseline and the proposed models even after training. Under 2 images per class (5 for inference similarity), the best scores are achieved right after initialization, and training only degrades performances. Building the best possible initialization is thus crucial in such cases.

## A.5 Conclusion and Perspectives

This chapter addresses transfer learning in an image classification context. In particular, alternative approaches were studied to re-use the knowledge inherent within the original pre-trained deep network in the target one (handling new image classes). Rather than only transferring network weights corresponding to the feature

extraction part, several initialization strategies were investigated to re-use and combine specifically identified weights from the pre-trained classifier into the target model. To validate the impact of the proposed method, three different similarity estimators were presented, one visual and two semantics, optimized the models across the different types of target classes, and monitored them while reducing the amount of data.

In the end, the proposed method produced systematically better initializations, faster trainings, and significantly superior long term performances in limited training data configurations. The consistency with which the best model, based on visual similarities, outperforms the baseline across the different types of target classes and amounts of data, along with its computational lightness (a few supplementary inferences in the network) suggest that it can be systematically adopted when performing transfer learning in this context.

# Bibliography

- [Ahn *et al.* 2019] Chanho Ahn, Eunwoo Kim and Songhwai Oh. *Deep Elastic Networks With Model Selection for Multi-Task Learning*. In Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV), 2019. (Cited on page [14](#).)
- [Azizpour *et al.* 2016] H. Azizpour, A. Razavian, J. Sullivan, A. Maki and S. Carlsson. *Factors of transferability for a generic convnet representation*. IEEE transactions on pattern analysis and machine intelligence, vol. 38, no. 9, pages 1790–1802, 2016. (Cited on page [86](#).)
- [Badrinarayanan *et al.* 2017] Vijay Badrinarayanan, Alex Kendall and Roberto Cipolla. *SegNet: A Deep Convolutional Encoder-Decoder Architecture for Image Segmentation*. IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 39, no. 12, pages 2481–2495, 2017. (Cited on page [34](#).)
- [Baxter 2000] Jonathan Baxter. *A Model of Inductive Bias Learning*. Journal of Artificial Intelligence Research, vol. 12, pages 149–198, 2000. (Cited on pages [1](#) and [2](#).)
- [Beckwith *et al.* 1991] R. Beckwith, C. Fellbaum, D. Gross and G. Miller. *WordNet: A lexical database organized on psycholinguistic principles*. Lexical acquisition: Exploiting on-line resources to build a lexicon, pages 211–232, 1991. (Cited on page [88](#).)
- [Bengio *et al.* 2013] Yoshua Bengio, Nicholas Léonard and Aaron C. Courville. *Estimating or Propagating Gradients Through Stochastic Neurons for Conditional Computation*. CoRR, vol. abs/1308.3432, 2013. (Cited on page [14](#).)
- [Bingel & Søgaard 2017] Joachim Bingel and Anders Søgaard. *Identifying beneficial task relations for multi-task learning in deep neural networks*. In Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics (EACL), pages 164–169, 2017. (Cited on pages [11](#) and [12](#).)
- [Bragman *et al.* 2019] Felix J.S. Bragman, Ryutaro Tanno, Sebastien Ourselin, Daniel C. Alexander and Jorge Cardoso. *Stochastic Filter Groups for Multi-Task CNNs: Learning Specialist and Generalist Convolution Kernels*. In The IEEE International Conference on Computer Vision (ICCV), pages 1385–1394, 2019. (Cited on pages [11](#), [14](#), [18](#), [19](#), [22](#), [23](#), [28](#), [42](#), [44](#) and [45](#).)
- [Capelle *et al.* 2012] M. Capelle, F. Frasincar, M. Moerland and F. Hogenboom. *Semantics-based news recommendation*. In Proceedings of the 2nd international conference on web intelligence, mining and semantics, page 27. ACM, 2012. (Cited on pages [88](#) and [90](#).)

- [Caruana 1997] Rich Caruana. *Multitask Learning*. Machine Learning, vol. 28, no. 1, pages 41–75, 1997. (Cited on pages 2, 5, 7, 15, 16, 24 and 41.)
- [Chaudhry *et al.* 2019] Arslan Chaudhry, Marc’Aurelio Ranzato, Marcus Rohrbach and Mohamed Elhoseiny. *Efficient Lifelong Learning with A-GEM*. In International Conference on Learning Representations, 2019. (Cited on pages 4 and 17.)
- [Chen *et al.* 2017] Liang-Chieh Chen, G. Papandreou, Florian Schroff and Hartwig Adam. *Rethinking Atrous Convolution for Semantic Image Segmentation*. ArXiv, vol. abs/1706.05587, 2017. (Cited on page 64.)
- [Chen *et al.* 2018] Zhao Chen, Vijay Badrinarayanan, Chen-Yu Lee and Andrew Rabinovich. *GradNorm: Gradient Normalization for Adaptive Loss Balancing in Deep Multitask Networks*. In Proceedings of the 35th International Conference on Machine Learning, volume 80, pages 794–803, 2018. (Cited on pages 6, 11, 14, 15, 16, 22, 29, 34, 42 and 47.)
- [Chen *et al.* 2020] Zhao Chen, Jiquan Ngiam, Yanping Huang, Thang Luong, Henrik Kretzschmar, Yuning Chai and Dragomir Anguelov. *Just Pick a Sign: Optimizing Deep Multitask Models with Gradient Sign Dropout*. Advances in Neural Information Processing Systems, vol. 33, pages 2039–2050, 2020. (Cited on pages 14, 17, 18, 42, 47 and 50.)
- [Chu *et al.* 2016] B. Chu, V. Madhavan, O. Beijbom, J. Hoffman and T. Darrell. *Best practices for fine-tuning visual classifiers to new domains*. In ECCV, pages 435–442. Springer, 2016. (Cited on pages 3, 86, 88 and 90.)
- [Collobert & Weston 2008] Ronan Collobert and Jason Weston. *A Unified Architecture for Natural Language Processing: Deep Neural Networks with Multitask Learning*. In Proceedings of the 25th International Conference on Machine Learning, ICML ’08, page 160–167, New York, NY, USA, 2008. Association for Computing Machinery. (Cited on page 5.)
- [Cordts *et al.* 2016] Marius Cordts, Mohamed Omran, Sebastian Ramos, Timo Rehfeld, Markus Enzweiler, Rodrigo Benenson, Uwe Franke, Stefan Roth and Bernt Schiele. *The Cityscapes Dataset for Semantic Urban Scene Understanding*. In The IEEE Conference on Computer Vision and Pattern Recognition (CVPR), pages 3213–3223, 2016. (Cited on pages 28, 47 and 49.)
- [Crawshaw 2020] Michael Crawshaw. *Multi-Task Learning with Deep Neural Networks: A Survey*. arXiv:2009.09796 [cs, stat], 2020. (Cited on pages 5, 6, 17 and 42.)
- [Dang *et al.* 2021] Vien Ngoc Dang, Giuseppe Di Giacomo, Viola Marconetto, Praateek Mathur, Rosa Cortese, Marco Lorenzi, Ferran Prados and Maria A.

- Zuluaga. *Vessel-CAPTCHA: an efficient learning framework for vessel annotation and segmentation*. CoRR, vol. abs/2101.09321, 2021. (Cited on page 2.)
- [Deng *et al.* 2009] J. Deng, W. Dong, R. Socher, L. Li, K. Li and L. Fei-Fei. *Imagenet: A large-scale hierarchical image database*. In CVPR, pages 248–255. Ieee, 2009. (Cited on pages 1, 4, 64 and 73.)
- [Doersch & Zisserman 2017] Carl Doersch and Andrew Zisserman. *Multi-Task Self-Supervised Visual Learning*. In Proceedings of the IEEE International Conference on Computer Vision (ICCV), 2017. (Cited on pages 11 and 12.)
- [Dwivedi & Roig 2019] Kshitij Dwivedi and Gemma Roig. *Representation Similarity Analysis for Efficient Task Taxonomy & Transfer Learning*. In Proceedings of the Conference on Computer Vision and Pattern Recognition (CVPR), 2019. (Cited on page 12.)
- [Désidéri 2012] Jean-Antoine Désidéri. *Multiple-gradient Descent Algorithm (MGDA) for Multiobjective Optimization*. Comptes Rendus Mathématique, vol. 350, no. 5, pages 313–318, 2012. (Cited on pages 16 and 42.)
- [Elsken *et al.* 2019] Thomas Elsken, Jan Hendrik Metzen and Frank Hutter. *Neural Architecture Search: A Survey*. Journal of Machine Learning Research, vol. 20, no. 55, pages 1–21, 2019. (Cited on page 13.)
- [Fernando *et al.* 2017] Chrisantha Fernando, Dylan Banarse, Charles Blundell, Yori Zwols, David Ha, Andrei A. Rusu, Alexander Pritzel and Daan Wierstra. *PathNet: Evolution Channels Gradient Descent in Super Neural Networks*. CoRR, vol. abs/1701.08734, 2017. (Cited on page 14.)
- [Gao *et al.* 2019] Yuan Gao, Jiayi Ma, Mingbo Zhao, Wei Liu and Alan L. Yuille. *NDDR-CNN: Layerwise Feature Fusing in Multi-Task CNNs by Neural Discriminative Dimensionality Reduction*. In The IEEE Conference on Computer Vision and Pattern Recognition (CVPR), pages 3200–3209, 2019. (Cited on pages 6, 13 and 21.)
- [Gao *et al.* 2020] Yuan Gao, Haoping Bai, Zequn Jie, Jiayi Ma, Kui Jia and Wei Liu. *MTL-NAS: Task-Agnostic Neural Architecture Search Towards General-Purpose Multi-Task Learning*. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), June 2020. (Cited on pages 13 and 14.)
- [Glorot & Bengio 2010] X. Glorot and Y. Bengio. *Understanding the difficulty of training deep feedforward neural networks*. In Proceedings of the thirteenth international conference on artificial intelligence and statistics, pages 249–256, 2010. (Cited on pages 86 and 92.)



- [Gomez *et al.* 2019] Aidan N. Gomez, Ivan Zhang, Siddhartha Rao Kamalakara, Divyam Madaan, Kevin Swersky, Yarin Gal and Geoffrey E. Hinton. *Learning Sparse Networks Using Targeted Dropout*. arXiv:1905.13678 [cs, stat], September 2019. (Cited on page 24.)
- [Gonzales Zuniga *et al.* 2018] Juan Diego Gonzales Zuniga, Thi-Lan-Anh Nguyen and Francois Bremond. *Residual Transfer Learning for Multiple Object Tracking*. In International Conference on Advanced Video and Signal-based Surveillance (AVSS), Auckland, New Zealand, November 2018. IEEE. (Cited on page 3.)
- [Guo *et al.* 2018] Michelle Guo, Albert Haque, De-An Huang, Serena Yeung and Li Fei-Fei. *Dynamic Task Prioritization for Multitask Learning*. pages 270–287, 2018. (Cited on pages 14, 15, 16 and 42.)
- [Hagiwara *et al.* 2018] Yuki Hagiwara, Joel En Wei Koh, Jen Hong Tan, Sulatha V. Bhandary, Augustinus Laude, Edward J. Ciaccio, Louis Tong and U. Rajendra Acharya. *Computer-aided diagnosis of glaucoma using fundus images: A review*. Computer Methods and Programs in Biomedicine, vol. 165, pages 1–12, 2018. (Cited on pages 62 and 64.)
- [Handler 2014] A. Handler. *An empirical study of semantic similarity in WordNet and Word2Vec*. 2014. (Cited on page 89.)
- [He *et al.* 2016a] K. He, X. Zhang, S. Ren and J. Sun. *Deep residual learning for image recognition*. In CVPR, pages 770–778, 2016. (Cited on page 87.)
- [He *et al.* 2016b] K. He, X. Zhang, S. Ren and J. Sun. *Identity mappings in deep residual networks*. In ECCV, pages 630–645. Springer, 2016. (Cited on page 87.)
- [He *et al.* 2016c] Kaiming He, Xiangyu Zhang, Shaoqing Ren and Jian Sun. *Deep Residual Learning for Image Recognition*. In The IEEE Conference on Computer Vision and Pattern Recognition (CVPR), pages 770–778, 2016. (Cited on pages 29 and 64.)
- [He *et al.* 2017] Kaiming He, Georgia Gkioxari, Piotr Dollar and Ross Girshick. *Mask R-CNN*. In The IEEE International Conference on Computer Vision (ICCV), pages 2961–2969, 2017. (Cited on pages 13, 21, 25, 64 and 65.)
- [Hinton & Salakhutdinov 2006] G. Hinton and R. Salakhutdinov. *Reducing the dimensionality of data with neural networks*. science, vol. 313, no. 5786, pages 504–507, 2006. (Cited on pages 3, 86 and 87.)
- [Hu *et al.* 2018] Jie Hu, Li Shen and Gang Sun. *Squeeze-and-Excitation Networks*. In The IEEE Conference on Computer Vision and Pattern Recognition (CVPR), pages 7132–7141, 2018. (Cited on page 19.)

- [Huh *et al.* 2016] M. Huh, P. Agrawal and A. Efros. *What makes ImageNet good for transfer learning?* arXiv preprint arXiv:1608.08614, 2016. (Cited on page 87.)
- [Huisman *et al.* 2021] Mike Huisman, Jan van Rijn and Aske Plaat. *A survey of deep meta-learning*. Artificial Intelligence Review, vol. 54, 08 2021. (Cited on page 13.)
- [Jang *et al.* 2017] Eric Jang, Shixiang Gu and Ben Poole. *Categorical Reparameterization with Gumbel-Softmax*. arXiv:1611.01144 [cs, stat], 2017. (Cited on pages 14 and 19.)
- [Jiang & Conrath 1997] J. J Jiang and D. Conrath. *Semantic similarity based on corpus statistics and lexical taxonomy*. arXiv preprint cmp-lg/9709008, 1997. (Cited on page 88.)
- [Kaisa 1999] Miettinen Kaisa. Nonlinear multiobjective optimization, volume 12 of *International Series in Operations Research & Management Science*. Kluwer Academic Publishers, Boston, USA, 1999. (Cited on page 16.)
- [Kendall *et al.* 2018] Alex Kendall, Yarin Gal and Roberto Cipolla. *Multi-Task Learning Using Uncertainty to Weigh Losses for Scene Geometry and Semantics*. In The IEEE Conference on Computer Vision and Pattern Recognition (CVPR), pages 7482–7491, 2018. (Cited on pages 7, 14, 15, 22 and 42.)
- [Kingma & Ba 2017] Diederik P. Kingma and Jimmy Ba. *Adam: A Method for Stochastic Optimization*. arXiv:1412.6980 [cs], 2017. (Cited on pages 29, 34, 45 and 46.)
- [Kirkpatrick *et al.* 2017] James Kirkpatrick, Razvan Pascanu, Neil Rabinowitz, Joel Veness, Guillaume Desjardins, Andrei A. Rusu, Kieran Milan, John Quan, Tiago Ramalho, Agnieszka Grabska-Barwinska, Demis Hassabis, Claudia Clopath, Dharshan Kumaran and Raia Hadsell. *Overcoming catastrophic forgetting in neural networks*, 2017. (Cited on page 4.)
- [Kokkinos 2017] Iasonas Kokkinos. *Ubertnet: Training a Universal Convolutional Neural Network for Low-, Mid-, and High-Level Vision Using Diverse Datasets and Limited Memory*. In The IEEE Conference on Computer Vision and Pattern Recognition (CVPR), pages 6129–6138, 2017. (Cited on pages 11 and 21.)
- [Konečný *et al.* 2015] Jakub Konečný, Brendan McMahan and Daniel Ramage. *Federated Optimization: Distributed Optimization Beyond the Datacenter*. arXiv:1511.03575 [cs.LG], 2015. (Cited on pages 44 and 53.)
- [Kriegeskorte *et al.* 2008] Nikolaus Kriegeskorte, Marieke Mur and Peter Bandettini. *Representational Similarity Analysis – Connecting the Branches of Systems Neuroscience*. Frontiers in systems neuroscience, page 4, 2008. (Cited on page 12.)

- [Krizhevsky *et al.* 2012] Alex Krizhevsky, Ilya Sutskever and Geoffrey E Hinton. *ImageNet Classification with Deep Convolutional Neural Networks*. In Advances in Neural Information Processing Systems 25, pages 1097–1105. 2012. (Cited on pages 1, 85 and 87.)
- [Leacock & Chodorow 1998] C. Leacock and M. Chodorow. *WordNet: An Electronic Lexical Database, chapter Combining Local Context and WordNet Similarity for Word Sense Identification, pages 265–283*, 1998. (Cited on page 88.)
- [Li *et al.* 2018] Hao Li, Zheng Xu, Gavin Taylor, Christoph Studer and Tom Goldstein. *Visualizing the Loss Landscape of Neural Nets*. In S. Bengio, H. Wallach, H. Larochelle, K. Grauman, N. Cesa-Bianchi and R. Garnett, editors, Advances in Neural Information Processing Systems, volume 31. Curran Associates, Inc., 2018. (Cited on page 7.)
- [Li *et al.* 2020] Xiang Li, Kaixuan Huang, Wenhao Yang, Shusen Wang and Zhihua Zhang. *On the Convergence of FedAvg on Non-IID Data*. arXiv:1907.02189 [stat.ML], 2020. (Cited on pages 44, 53 and 54.)
- [Liang *et al.* 2018] Jason Liang, Elliot Meyerson and Risto Miikkulainen. *Evolutionary Architecture Search for Deep Multitask Networks*. In Proceedings of the Genetic and Evolutionary Computation Conference, GECCO '18, page 466–473, New York, NY, USA, 2018. Association for Computing Machinery. (Cited on pages 13 and 14.)
- [Lin *et al.* 1998] D. Lin *et al.* *An information-theoretic definition of similarity*. In Icml, volume 98, pages 296–304. Citeseer, 1998. (Cited on page 88.)
- [Lin *et al.* 2017] Tsung-Yi Lin, Priya Goyal, Ross Girshick, Kaiming He and Piotr Dollár. *Focal loss for dense object detection*. In Proceedings of the IEEE international conference on computer vision, pages 2980–2988, 2017. (Cited on page 67.)
- [Lin *et al.* 2019] Xi Lin, Hui-Ling Zhen, Zhenhua Li, Qing-Fu Zhang and Sam Kwong. *Pareto Multi-Task Learning*. In Advances in Neural Information Processing Systems 32, pages 12060–12070, 2019. (Cited on pages 14, 16, 17 and 42.)
- [Liu *et al.* 2015a] Xiaodong Liu, Jianfeng Gao, Xiaodong He, Li Deng, Kevin Duh and Ye-yi Wang. *Representation Learning Using Multi-Task Deep Neural Networks for Semantic Classification and Information Retrieval*. In Proceedings of the 2015 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, pages 912–921, Denver, Colorado, May–June 2015. Association for Computational Linguistics. (Cited on page 5.)
- [Liu *et al.* 2015b] Ziwei Liu, Ping Luo, Xiaogang Wang and Xiaoou Tang. *Deep Learning Face Attributes in the Wild*. In The IEEE International Conference

- on Computer Vision (ICCV), pages 3730–3738, 2015. (Cited on pages 29, 47 and 50.)
- [Liu *et al.* 2019a] Shengchao Liu, Yingyu Liang and Anthony Gitter. *Loss-Balanced Task Weighting to Reduce Negative Transfer in Multi-Task Learning*. Proceedings of the AAAI Conference on Artificial Intelligence, vol. 33, no. 01, pages 9977–9978, Jul. 2019. (Cited on pages 14 and 15.)
- [Liu *et al.* 2019b] Shikun Liu, Edward Johns and Andrew J. Davison. *End-To-End Multi-Task Learning With Attention*. In The IEEE Conference on Computer Vision and Pattern Recognition (CVPR), pages 1871–1880, 2019. (Cited on pages 13, 14, 15, 21, 22, 28, 34 and 42.)
- [Lopez-Paz & Ranzato 2017] David Lopez-Paz and Marc’Aurelio Ranzato. *Gradient Episodic Memory for Continual Learning*. Advances in Neural Information Processing Systems, vol. 30, 2017. (Cited on pages 4 and 17.)
- [Lu *et al.* 2017a] Yongxi Lu, Abhishek Kumar, Shuangfei Zhai, Yu Cheng, Tara Javidi and Rogerio Feris. *Fully-Adaptive Feature Sharing in Multi-Task Networks With Applications in Person Attribute Classification*. In The IEEE Conference on Computer Vision and Pattern Recognition (CVPR), pages 5334–5343, 2017. (Cited on pages 13 and 21.)
- [Lu *et al.* 2017b] Yongxi Lu, Abhishek Kumar, Shuangfei Zhai, Yu Cheng, Tara Javidi and Rogerio Feris. *Fully-Adaptive Feature Sharing in Multi-Task Networks With Applications in Person Attribute Classification*. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR), July 2017. (Cited on page 14.)
- [Maddison *et al.* 2017] Chris J. Maddison, Andriy Mnih and Yee Whye Teh. *The Concrete Distribution: A Continuous Relaxation of Discrete Random Variables*. arXiv:1611.00712 [cs, stat], 2017. (Cited on pages 14 and 19.)
- [Mallya *et al.* 2018] Arun Mallya, Dillon Davis and Svetlana Lazebnik. *Piggyback: Adapting a Single Network to Multiple Tasks by Learning to Mask Weights*. In Proceedings of the European Conference on Computer Vision (ECCV), pages 67–82, September 2018. (Cited on pages 4, 14, 18 and 19.)
- [Mancini *et al.* 2018] Massimiliano Mancini, Elisa Ricci, Barbara Caputo and Samuel Rota Buló. *Adding New Tasks to a Single Network with Weight Transformations using Binary Masks*. In Proceedings of the European Conference on Computer Vision (ECCV) Workshops, September 2018. (Cited on pages 4, 14, 18 and 19.)
- [Maninis *et al.* 2019] Kevis-Kokitsi Maninis, Ilija Radosavovic and Iasonas Kokkinos. *Attentive Single-Tasking of Multiple Tasks*. In 2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), pages 1851–1860, 2019. (Cited on pages 7, 11, 14, 18, 19, 20, 22, 23, 24, 28, 29, 34, 42, 44 and 45.)

- [Martinez Alonso & Plank 2017] Hector Martinez Alonso and Barbara Plank. *When is multitask learning effective? Semantic sequence prediction under varying data conditions*. In Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics (EACL), pages 1–10, 2017. (Cited on pages 11 and 12.)
- [Meyerson & Miikkulainen 2017] Elliot Meyerson and Risto Miikkulainen. *Beyond Shared Hierarchies: Deep Multitask Learning through Soft Layer Ordering*. CoRR, vol. abs/1711.00108, 2017. (Cited on pages 11, 13 and 14.)
- [Mikolov *et al.* 2013] T. Mikolov, K. Chen, G. Corrado and J. Dean. *Efficient estimation of word representations in vector space*. arXiv preprint arXiv:1301.3781, 2013. (Cited on page 89.)
- [Misra *et al.* 2016] Ishan Misra, Abhinav Shrivastava, Abhinav Gupta and Martial Hebert. *Cross-Stitch Networks for Multi-Task Learning*. In The IEEE Conference on Computer Vision and Pattern Recognition (CVPR), pages 3994–4003, 2016. (Cited on pages 6, 11, 13, 14 and 21.)
- [Mordan *et al.* 2018] Taylor Mordan, Nicolas Thome, Gilles Henaff and Matthieu Cord. *Revisiting Multi-Task Learning with ROCK: a Deep Residual Auxiliary Block for Visual Detection*. In Advances in Neural Information Processing Systems 31, pages 1310–1322, 2018. (Cited on pages 13 and 21.)
- [Orlando *et al.* 2020] José Ignacio Orlando, Huazhu Fu, João Barbosa Breda, Karel van Keer, Deepti R. Bathula, Andrés Diaz-Pinto, Ruogu Fang, Pheng-Ann Heng, Jeyoung Kim, JoonHo Lee, Joonseok Lee, Xiaoxiao Li, Peng Liu, Shuai Lu, Balamurali Murugesan, Valery Naranjo, Sai Samarth R. Phaye, Sharath M. Shankaranarayana, Apoorva Sikka, Jaemin Son, Anton van den Hengel, Shujun Wang, Junyan Wu, Zifeng Wu, Guanghui Xu, Yongli Xu, Pengshuai Yin, Fei Li, Xiulan Zhang, Yanwu Xu and Hrvoje Bogunović. *REFUGE Challenge: A unified framework for evaluating automated methods for glaucoma assessment from fundus photographs*. Medical Image Analysis, vol. 59, page 101570, 2020. (Cited on pages 62, 63, 64 and 66.)
- [Parisi *et al.* 2019] German I. Parisi, Ronald Kemker, Jose L. Part, Christopher Kanan and Stefan Wermter. *Continual lifelong learning with neural networks: A review*. Neural Networks, vol. 113, pages 54–71, 2019. (Cited on pages 4 and 5.)
- [Pascal *et al.* 2019] Lucas Pascal, Xavier Bost and Benoit Huet. *Semantic and Visual Similarities for Efficient Knowledge Transfer in CNN Training*. In 2019 International Conference on Content-Based Multimedia Indexing (CBMI), pages 1–6, 2019. (Cited on page 85.)
- [Pascal *et al.* 2021a] Lucas Pascal, Pietro Michiardi, Xavier Bost, Benoit Huet and Maria A. Zuluaga. *Maximum Roaming Multi-Task Learning*. Proceedings

- of the AAAI Conference on Artificial Intelligence, vol. 35, no. 10, pages 9331–9341, May 2021. (Cited on pages [21](#), [42](#), [44](#), [45](#) and [47](#).)
- [Pascal *et al.* 2021b] Lucas Pascal, Pietro Michiardi, Xavier Bost, Benoit Huet and Maria A. Zuluaga. *Optimization Strategies in Multi-Task Learning: Averaged or Independent Losses?* arxiv:2109.11678 [cs], 2021. (Cited on page [41](#).)
- [Pinto & Gupta 2017] Lerrel Pinto and Abhinav Gupta. *Learning to push by grasping: Using multiple tasks for effective learning*. In 2017 IEEE International Conference on Robotics and Automation (ICRA), pages 2161–2168, 2017. (Cited on page [5](#).)
- [Pratt 1993] L. Y. Pratt. *Discriminability-Based Transfer between Neural Networks*. In S. Hanson, J. Cowan and C. Giles, editors, Advances in Neural Information Processing Systems, volume 5. Morgan-Kaufmann, 1993. (Cited on page [3](#).)
- [Rebuffi *et al.* 2018] Sylvestre-Alvise Rebuffi, Hakan Bilen and Andrea Vedaldi. *Efficient Parametrization of Multi-Domain Deep Neural Networks*. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR), June 2018. (Cited on page [19](#).)
- [Resnik 1995] P. Resnik. *Using information content to evaluate semantic similarity in a taxonomy*. arXiv preprint cmp-lg/9511007, 1995. (Cited on page [88](#).)
- [Ronneberger *et al.* 2015a] Olaf Ronneberger, Philipp Fischer and Thomas Brox. *U-Net: Convolutional Networks for Biomedical Image Segmentation*. In Nassir Navab, Joachim Hornegger, William M. Wells and Alejandro F. Frangi, editors, Medical Image Computing and Computer-Assisted Intervention – MICCAI 2015, Lecture Notes in Computer Science, pages 234–241, Cham, 2015. Springer International Publishing. (Cited on pages [47](#) and [64](#).)
- [Ronneberger *et al.* 2015b] Olaf Ronneberger, Philipp Fischer and Thomas Brox. *U-Net: Convolutional Networks for Biomedical Image Segmentation*. arxiv:1505.04597 [cs], 2015. (Cited on page [65](#).)
- [Rosenbaum *et al.* 2017] Clemens Rosenbaum, Tim Klinger and Matthew Riemer. *Routing Networks: Adaptive Selection of Non-linear Functions for Multi-Task Learning*. CoRR, vol. abs/1711.01239, 2017. (Cited on pages [11](#) and [14](#).)
- [Ruder 2017] Sebastian Ruder. *An Overview of Multi-Task Learning in Deep Neural Networks*. arXiv:1706.05098 [cs, stat], 2017. (Cited on pages [5](#) and [6](#).)
- [Russakovsky *et al.* 2015] O. Russakovsky, J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma, Z. Huang, A. Karpathy, A. Khosla, M. Bernstein *et al.* *Imagenet large scale visual recognition challenge*. IJCV, vol. 115, no. 3, pages 211–252, 2015. (Cited on pages [85](#), [87](#) and [91](#).)



- [Rusu *et al.* 2016] Andrei A. Rusu, Neil C. Rabinowitz, Guillaume Desjardins, Hubert Soyer, James Kirkpatrick, Koray Kavukcuoglu, Razvan Pascanu and Raia Hadsell. *Progressive Neural Networks*, 2016. (Cited on page 4.)
- [Safadi *et al.* 2014] B. Safadi, M. Sahuguet and B. Huet. *When textual and visual information join forces for multimedia retrieval*. In ICMR, 2014. (Cited on page 95.)
- [Sener & Koltun 2018] Ozan Sener and Vladlen Koltun. *Multi-Task Learning as Multi-Objective Optimization*. In Advances in Neural Information Processing Systems 31, pages 527–538, 2018. (Cited on pages 11, 14, 16, 17, 22, 29, 42 and 47.)
- [Shokri & Shmatikov 2015] Reza Shokri and Vitaly Shmatikov. *Privacy-Preserving Deep Learning*. In Proceedings of the 22nd ACM SIGSAC Conference on Computer and Communications Security, page 1310–1321, 2015. (Cited on page 44.)
- [Shorten & Khoshgoftaar 2019] Connor Shorten and T. Khoshgoftaar. *A survey on Image Data Augmentation for Deep Learning*. Journal of Big Data, vol. 6, pages 1–48, 2019. (Cited on page 2.)
- [Silberman *et al.* 2012] Nathan Silberman, Derek Hoiem, Pushmeet Kohli and Rob Fergus. *Indoor Segmentation and Support Inference from RGBD Images*. In European Conference on Computer Vision (ECCV) 2012, Lecture Notes in Computer Science, pages 746–760, 2012. (Cited on pages 13, 28 and 47.)
- [Simonyan & Zisserman 2015] Karen Simonyan and Andrew Zisserman. *Very Deep Convolutional Networks for Large-Scale Image Recognition*. arXiv:1409.1556 [cs], 2015. (Cited on pages 65 and 73.)
- [Singh 1992] Satinder Pal Singh. *Transfer of Learning by Composing Solutions of Elemental Sequential Tasks*. Machine Learning, vol. 8, no. 3-4, pages 323–339, 1992. (Cited on page 25.)
- [Sinha *et al.* 2018] Ayan Sinha, Zhao Chen, Vijay Badrinarayanan and Andrew Rabinovich. *Gradient Adversarial Training of Neural Networks*. arXiv:1806.08028 [cs, stat], 2018. (Cited on pages 14, 15, 16, 19 and 22.)
- [Song *et al.* 2019] Jie Song, Yixin Chen, Xinchao Wang, Chengchao Shen and Mingli Song. *Deep Model Transferability from Attribution Maps*. In Advances in Neural Information Processing Systems (NeurIPS), pages 6179–6189, 2019. (Cited on page 12.)
- [Srivastava *et al.* 2014] Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever and Ruslan Salakhutdinov. *Dropout: A Simple Way to Prevent Neural Networks from Overfitting*. Journal of Machine Learning Research, vol. 15, no. 56, pages 1929–1958, 2014. (Cited on pages 2, 23 and 25.)

- [Standley *et al.* 2020] Trevor Standley, Amir Zamir, Dawn Chen, Leonidas Guibas, Jitendra Malik and Silvio Savarese. *Which Tasks Should Be Learned Together in Multi-task Learning?* In Proceedings of the 37th International Conference on Machine Learning, volume 119, pages 9120–9132, 2020. (Cited on pages 11 and 12.)
- [Strezoski *et al.* 2019a] Gjorgji Strezoski, Nanne van Noord and Marcel Worring. *Many Task Learning With Task Routing*. In The IEEE International Conference on Computer Vision (ICCV), pages 1375–1384, 2019. (Cited on pages 7, 14, 18, 19, 22, 23, 24, 28, 29, 34, 42 and 44.)
- [Strezoski *et al.* 2019b] Gjorgji Strezoski, Nanne van Noord and Marcel Worring. *Learning Task Relatedness in Multi-Task Learning for Images in Context*. In Proceedings of the 2019 on International Conference on Multimedia Retrieval, page 78–86, 2019. (Cited on pages 6, 11, 12, 13 and 14.)
- [Sun *et al.* 2019] Ximeng Sun, Rameswar Panda and Rog rio Schmidt Feris. *AdaShare: Learning What To Share For Efficient Deep Multi-Task Learning*. CoRR, vol. abs/1911.12423, 2019. (Cited on page 14.)
- [Szegedy *et al.* 2015] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke and A. Rabinovich. *Going deeper with convolutions*. In CVPR, pages 1–9, 2015. (Cited on page 87.)
- [Szegedy *et al.* 2017] C. Szegedy, S. Ioffe, V. Vanhoucke and A. Alemi. *Inception-v4 Inception-Resnet and the impact of residual connections on learning*. In AAAI, 2017. (Cited on pages 64 and 87.)
- [Tajbakhsh *et al.* 2016] Nima Tajbakhsh, Jae Y. Shin, Suryakanth R. Gurudu, R. Todd Hurst, Christopher B. Kendall, Michael B. Gotway and Jianming Liang. *Convolutional Neural Networks for Medical Image Analysis: Full Training or Fine Tuning?* IEEE Transactions on Medical Imaging, vol. 35, no. 5, pages 1299–1312, 2016. (Cited on pages 25 and 62.)
- [Tajbakhsh *et al.* 2020] Nima Tajbakhsh, Laura Jeyaseelan, Qian Li, Jeffrey N. Chiang, Zhihao Wu and Xiaowei Ding. *Embracing imperfect datasets: A review of deep learning solutions for medical image segmentation*. Medical Image Analysis, vol. 63, page 101693, 2020. (Cited on pages 2, 4 and 73.)
- [Tamaazousti *et al.* 2017] Y. Tamaazousti, H. Le Borgne, C. Hudelot, M. Seddik and M. Tamaazousti. *Learning More Universal Representations for Transfer-Learning*. arXiv preprint arXiv:1712.09708, 2017. (Cited on pages 75, 86, 88 and 90.)
- [Tham *et al.* 2014] YC Tham, X Li, TY Wong, HA Quigley, T Aung and CY Cheng. *Global prevalence of glaucoma and projections of glaucoma burden through 2040: a systematic review and meta-analysis*. Ophthalmology 121, pages 2081–2090, 2014. (Cited on page 61.)



- [Thrun & Pratt 1998] Sebastian Thrun and Lorien Pratt. Learning to learn: Introduction and overview, pages 3–17. Springer US, Boston, MA, 1998. (Cited on page 2.)
- [Vandenhende *et al.* 2020] Simon Vandenhende, Stamatios Georgoulis and Luc Van Gool. *MTI-Net: Multi-Scale Task Interaction Networks for Multi-Task Learning*. arXiv:2001.06902 [cs], 2020. (Cited on pages 13 and 22.)
- [Vandenhende *et al.* 2021] Simon Vandenhende, Stamatios Georgoulis, Wouter Van Gansbeke, Marc Proesmans, Dengxin Dai and Luc Van Gool. *Multi-Task Learning for Dense Prediction Tasks: A Survey*. IEEE Transactions on Pattern Analysis and Machine Intelligence, pages 1–1, 2021. (Cited on pages 5, 6 and 42.)
- [Wang *et al.* 2016] J. Wang, Y. Yang, J. Mao, Z. Huang, C. Huang and W. Xu. *Cnn-rnn: A unified framework for multi-label image classification*. In CVPR, pages 2285–2294, 2016. (Cited on page 90.)
- [Wang *et al.* 2017] Y. Wang, D. Ramanan and M. Hebert. *Growing a brain: Fine-tuning by increasing model capacity*. In CVPR, pages 2471–2480, 2017. (Cited on pages 3, 86 and 88.)
- [Wang 2014] H. Wang. *Introduction to Word2vec and its application to find predominant word senses*. 2014. (Cited on page 89.)
- [Wu & Palmer 1994] Z. Wu and M. Palmer. *Verbs semantics and lexical selection*. In Proceedings of the 32nd annual meeting on Association for Computational Linguistics, pages 133–138. Association for Computational Linguistics, 1994. (Cited on pages 88 and 90.)
- [Xu *et al.* 2018] Dan Xu, Wanli Ouyang, Xiaogang Wang and Nicu Sebe. *PAD-Net: Multi-Tasks Guided Prediction-and-Distillation Network for Simultaneous Depth Estimation and Scene Parsing*. In The IEEE Conference on Computer Vision and Pattern Recognition (CVPR), pages 675–684, 2018. (Cited on pages 13 and 22.)
- [Yosinski *et al.* 2014] Jason Yosinski, Jeff Clune, Yoshua Bengio and Hod Lipson. *How transferable are features in deep neural networks?* In Z. Ghahramani, M. Welling, C. Cortes, N. D. Lawrence and K. Q. Weinberger, editors, Advances in Neural Information Processing Systems 27, pages 3320–3328. Curran Associates, Inc., 2014. (Cited on pages 3, 25, 37, 75, 86, 87, 88 and 90.)
- [Yu *et al.* 2020] Tianhe Yu, Saurabh Kumar, Abhishek Gupta, Sergey Levine, Karol Hausman and Chelsea Finn. *Gradient Surgery for Multi-Task Learning*. Advances in Neural Information Processing Systems, vol. 33, pages 5824–5836, 2020. (Cited on pages 5, 11, 14, 17 and 42.)

- [Zamir *et al.* 2018] Amir R. Zamir, Alexander Sax, William Shen, Leonidas J. Guibas, Jitendra Malik and Silvio Savarese. *Taskonomy: Disentangling Task Transfer Learning*. In The IEEE Conference on Computer Vision and Pattern Recognition (CVPR), pages 3712–3722, 2018. (Cited on pages 3, 11, 12, 25 and 86.)
- [Zenke *et al.* 2017] Friedemann Zenke, Ben Poole and Surya Ganguli. *Continual Learning Through Synaptic Intelligence*. In Doina Precup and Yee Whye Teh, editors, Proceedings of the 34th International Conference on Machine Learning, volume 70 of *Proceedings of Machine Learning Research*, pages 3987–3995. PMLR, 06–11 Aug 2017. (Cited on page 4.)
- [Zhang & Yang 2021] Yu Zhang and Qiang Yang. *A Survey on Multi-Task Learning*. IEEE Transactions on Knowledge and Data Engineering, pages 1–1, 2021. (Cited on page 5.)
- [Zhang *et al.* 2018] Yu Zhang, Ying Wei and Qiang Yang. *Learning to Multitask*. In Advances in Neural Information Processing Systems 31, pages 5771–5782, 2018. (Cited on page 22.)
- [Zhang *et al.* 2019] Zhenyu Zhang, Zhen Cui, Chunyan Xu, Yan Yan, Nicu Sebe and Jian Yang. *Pattern-Affinitive Propagation Across Depth, Surface Normal and Semantic Segmentation*. In The IEEE Conference on Computer Vision and Pattern Recognition (CVPR), pages 4106–4115, 2019. (Cited on pages 13 and 22.)
- [Zhao *et al.* 2019] Rongchang Zhao, Wangmin Liao, Beiji Zou, Zailiang Chen and Shuo Li. *Weakly-Supervised Simultaneous Evidence Identification and Segmentation for Automated Glaucoma Diagnosis*. Proceedings of the AAAI Conference on Artificial Intelligence, vol. 33, no. 01, pages 809–816, Jul. 2019. (Cited on page 64.)