



UNIVERSITÀ  
DEGLI STUDI  
FIRENZE

PHD PROGRAM IN SMART COMPUTING  
DIPARTIMENTO DI INGEGNERIA DELL'INFORMAZIONE (DINFO)

# OPENING MACHINE EYES OVER TIME: INPUT TUNING AND MOTION-DRIVEN LEARNING

**Simone Marullo**

Dissertation presented in partial fulfillment of the requirements  
for the degree of Doctor of Philosophy in Smart Computing

*PhD Program in Smart Computing*  
*University of Florence, University of Pisa, University of Siena*

# OPENING MACHINE EYES OVER TIME: INPUT TUNING AND MOTION-DRIVEN LEARNING

**Simone Marullo**

**Advisor:**

---

Prof. Marco Gori

**Head of the PhD Program:**

---

Prof. Stefano Berretti

**Evaluation Committee:**

Prof. Elisa Ricci, *University of Trento, Fondazione Bruno Kessler*  
Prof. Simone Calderara, *University of Modena and Reggio Emilia*

## Acknowledgments

Embarking on this research journey has been both challenging and enlightening, and I owe much of my growth and accomplishments to those who have supported and guided me along the way.

At the very forefront of these guiding lights is Prof. Marco Gori. His role as my PhD supervisor transcended mere academic guidance. His profound insights, unwavering belief in my potential, and continuous encouragement have shaped my research trajectory in countless ways. Thank you, Marco, for your mentorship and for inspiring me to always strive for excellence.

I am deeply indebted to Prof. Stefano Melacci, whose unwavering guidance and timely feedback on various research challenges have been indispensable. His mentorship has been a guiding light, shaping the course of this research.

Leading the way of my collaborations is Matteo Tiezzi. Not only has he provided invaluable contributions to this work, but he has also stood by me as one of my dearest friends, sharing triumph and despair of academic pursuit and personal milestones. Ranting about the experiments has been life-saving!

I want to give a big shoutout to my buddy Alessio Medaglini. We've been through the master's program together, stayed friends, and now we're both tackling our PhDs. Cheers, Alessio!

Special recognition goes to my fellow PhD colleagues: Lapo Faggi, Enrico Meloni, Michele Casoni, and Kamyar Zeinalipour for their camaraderie and insights. The SAILAB in Siena has been a haven of collaboration and shared experiences. Engaging with researchers like Andrea Zugarini, Francesco Giannini, Stefano Fioravanti, and Alessandro Betti, be it over academic discussions or shared lunches, has added a richness to my PhD journey that I will always cherish.

I would like to highlight the valuable time spent at KU Leuven's ESAT department in Belgium during the summer of 2022. Under the guidance of Prof. Tinne Tuytelaars, I was introduced to new perspectives and methodologies that have been instrumental in furthering my research. Additionally, my summer in 2023 at Carbonfarm Technologies in Paris presented unique challenges and learnings, offering a practical dimension to my academic endeavors. Beyond the walls of this innovative startup, I found a community of kind-hearted and welcoming individuals.

I owe a deep sense of gratitude to my Supervisory Committee. Prof. Fabio Roli and Prof. Marcello Pelillo, your critical insights and perspectives have significantly enriched my research path.

I wish to express sincere appreciation to Regione Toscana for their support through the Pegaso grant, which has contributed to funding of my PhD scholarship and greatly facilitated my research pursuits. If it wasn't for the intriguing 'obligation' of spending six months abroad, I must confess I might not have ventured out for

such an extended period. Yet, in a twist of fate (and grant stipulations), I find myself overwhelmingly grateful for this nudge out of my comfort zone. The journey not only broadened my professional horizons but gifted me with invaluable life experiences that I wouldn't trade for anything. Sometimes, a little obligation turns out to be the best kind of serendipity!

On a personal note, my deepest gratitude goes to my family. Their unwavering support, endless encouragement, and faith in my abilities have been the bedrock upon which this journey was built. In times of doubt and challenge, it was their belief in me that fueled my perseverance. I want to express the greatest possible acknowledgment to my older sister Sara, since it is probably because of her that I have had the great idea of starting a PhD and later I have managed persisting through it. It has been extremely useful to share my ideas and concerns with a brilliant physicist-engineer, who has always succeeded in compensating the lack of knowledge of the literature with intuition and willingness to help. My younger sister Caterina has substantially helped me with countless encouragements and stupid jokes—and can now hilariously mimic my conference presentations. My parents, to whom I owe my greatest sense of respect and admiration, have always supported me with kind gestures, with the right questions and all the unreasonable reasonableness typical of good parents.

Lastly, to all who have contributed directly or indirectly to this journey, your roles have not been minor. My heartfelt appreciation goes out to each one of you.

---

## Abstract

Vision is not just the biological ability to detect light; it is an essential part of the capability of animals, humans, and future machines to interpret, understand and act in the environment. If a 2-year-old child encounters their very first tractor while hearing its name, the child from that point forward will recognize all variety of tractors, without confusing them with cars or trucks. To date, this surprising talent in visual learning, acquired with such a limited supervision from external agents, is something not easily reproducible in computer vision. Inspired by the quest of achieving similar learning schemes, in this work we study several aspects of computer vision, proposing innovative neural network training techniques.

The first part of the thesis introduces the concept of input tuning for smooth learning paths, which involves dynamic transformations of inputs during training, inspired by the gradual visual skill acquisition observed in infants. We present a method that breaks down complex learning tasks into a series of incrementally challenging sub-tasks. This is achieved through input transformations that match the learner's skill level, enhancing model performance and deepening our understanding of the learning process. Then, we use the notion of input tuning in a different scenario, where a learner faces diverse tasks without a meaningful order, risking catastrophic forgetting. A novel training method keeps the learner's core static while using learnable transformations in the input space for environment adaptation, mitigating forgetting in realistic situations.

The second part of the thesis shifts from the supervised learning focus of the first part, aiming to create autonomous visual agents that learn directly from their surroundings without human intervention. These agents forgo large labelled data collections, observing continuous video streams and learning online, electing motion as the primary source of information. As such, we start by investigating optical flow estimation in dynamic environments, using a purely online unsupervised approach. We then present two self-supervised learning techniques. The first employs an attention trajectory, simulating human visual attention and allowing agents to establish semantic connections among pixels. The second is motion-based, resulting from a layered autonomous development process. Results indicate significant progress in the quest for autonomous visual skill development, with intriguing open directions.

Benefits obtained from controlling the learning pace through input tuning naturally open to future research directions, aimed at improving the robustness of visual agents that learn online without supervision.

# Contents

<b>Contents</b>	<b>1</b>
<b>1 Introduction</b>	<b>3</b>
1.1 Motivations and open issues . . . . .	5
1.2 Contributions . . . . .	8
1.3 Structure of the thesis . . . . .	9
<b>2 Background: A developmental approach to learning</b>	<b>10</b>
2.1 Human and Machine Developmental Learning . . . . .	10
2.2 Continual and Lifelong Machine Learning . . . . .	13
2.3 Visual Representation Learning . . . . .	18
<b>I Progressively learning in the input space</b>	<b>23</b>
<b>3 Input tuning for Friendly Training</b>	<b>26</b>
3.1 Introduction . . . . .	27
3.2 Relation with existing works . . . . .	29
3.3 Friendly Training: implementations . . . . .	31
3.4 Experiments . . . . .	38
3.5 Discussion . . . . .	43
<b>4 Input tuning for Continual Learning</b>	<b>47</b>
4.1 Introduction . . . . .	47
4.2 Relation with existing works . . . . .	49
4.3 Continually tuning a pre-trained model . . . . .	51
4.4 Experiments . . . . .	55
4.5 Discussion . . . . .	61
<b>II Growing visual agents from video streams</b>	<b>65</b>
<b>5 Optical flow estimation with online learning</b>	<b>68</b>

---

5.1	Introduction . . . . .	68
5.2	Relation with existing works . . . . .	70
5.3	Continual learning of optical flow estimation . . . . .	72
5.4	Experiments . . . . .	76
5.5	Discussion . . . . .	80
<b>6</b>	<b>Self-supervised online learning for autonomous visual agents</b>	<b>85</b>
6.1	Introduction . . . . .	86
6.2	Relation with existing works . . . . .	88
6.3	Learning and evaluation setting . . . . .	90
6.4	COAT: attention-based model . . . . .	93
6.5	CMOSFET: motion-based model . . . . .	98
6.6	Experiments and discussion . . . . .	109
<b>7</b>	<b>Conclusions and future works</b>	<b>119</b>
<b>A</b>	<b>Experimental details on optical flow estimation</b>	<b>124</b>
<b>B</b>	<b>Experimental details on attention-based agents (COAT)</b>	<b>130</b>
<b>C</b>	<b>Experimental details on motion-based agents (CMOSFET)</b>	<b>134</b>
<b>D</b>	<b>Publications</b>	<b>139</b>
	<b>Bibliography</b>	<b>141</b>

# Chapter 1

## Introduction

*We hope to be able to build a program that can learn, as a child does... instead of being spoon-fed the tremendous information necessary.*

— R. C. Schank, 1972

*The total act [...] may seem to be a functionally coherent unit of behavior; but it is constructed by a continual process from undifferentiated behavior, just as the sculptor shapes his figure from a lump of clay.*

— B.F. Skinner, 1953

*The frog does not seem to see or, at any rate, is not concerned with the detail of stationary parts of the world around him. He will starve to death surrounded by food if it is not moving.*

— J. Y. Lettvin et al., 1959

In the realm of artificial intelligence, a remarkable transformation has occurred, transitioning from a theoretical pursuit in academia to a practical, everyday application. In the year 2023, this evolution was underscored by the widespread adoption of powerful models, enabling sophisticated chatbots and innovative multimedia generation models among the others. These groundbreaking achievements have fostered a palpable sense of optimism pervading both research and industry sectors. Notably, market experts predict a robust growth rate exceeding 30% year-over-year in AI markets until at least 2030 (GrandViewResearch, 2022; NextMSC, 2023). The history of artificial intelligence, however, has been marked by a fluctuating narrative of optimism and skepticism, driven by an audacious yet ambiguously defined objective: creating computer programs capable of emulating innate human abilities such as language and vision (Parisi et al., 2019).



Actually, the research work discussed in this thesis concerns computer vision, that stands out as a cornerstone of research and industry applications in AI. Computer vision aims to empower computer systems to perceive and comprehend the visual world akin to human perception. In the 1960s, AI was a full-fledged academic discipline, sparking considerable optimism on computer vision as well. During this era, Seymour Papert, a professor at MIT's AI lab, initiated the Summer Vision Project (Papert, 1966), aiming to solve the machine vision problem in a matter of months with the cooperation of a small group of MIT students. Their mission was to develop a platform capable of automatically segmenting background from foreground and extracting distinct objects from real-world images. However, as the reader can likely guess, the students did not meet the deadline. More than fifty years later, computer vision remains a formidable challenge. Nevertheless, many consider the Summer Vision Project as the official birth of computer vision as a scientific field.

Over the years, the computer vision research community gradually began to diverge from the broader field of AI. This shift was driven by an increasing emphasis on highly specialized algorithms and meticulous feature engineering. One of the initial milestones in computer vision's journey was the development of the Viola-Jones object detection algorithm (Viola and Jones, 2001). This breakthrough not only saw widespread adoption in commercial products, such as its use in facial detection for digital cameras, but also continues to find applications in low-power devices to this day.

A pivotal development in computer vision and AI was the pursuit of representation learning, a paradigm where systems acquire knowledge autonomously by extracting data patterns instead of relying on pre-defined rules. This approach revolutionized the field, rendering traditional symbolic AI less popular while elevating machine learning, particularly through neural networks, as the predominant methodology for all kind of problems, even more so in computer vision. In this landscape, the focus shifted towards obtaining rich input representations, showcasing the monumental impact of representation learning on problem-solving within machine learning.

Significant breakthroughs, including seminal work by Lecun et al. (1998) and Krizhevsky et al. (2012), led to the establishment of deep convolutional neural networks (CNNs) as the preferred architecture for complex computer vision tasks. CNNs, distinguished by their convolutional layers that detect local spatial features in images, mimic certain processes in human vision (Lindsay, 2021). However, these advancements were largely facilitated by the unprecedented availability of data, exemplified by datasets like ImageNet (Russakovsky et al., 2015), which popularized the idea of massive data collection to advance machine learning performance. Recent experimental findings (Tolstikhin et al., 2021) suggest that, given copious amounts of data, non-convolutional architectures might excel in vision benchmarks,

highlighting the paramount role that data abundance has had and is having on the research efforts.

## 1.1 Motivations and open issues

Machine Learning methods have backed impressive results in Computer Vision, as well in many other domains, such as Natural Language Processing and Robotics. In spite of this incontrovertible fact, we will now briefly present some general issues affecting the common paradigm, along with an alternative vision of intelligent machines that inspired most of the research work presented in this thesis.

In the broader domain of Artificial Intelligence (AI), the trajectory of development has witnessed a striking divergence between two paradigms. Symbolic AI, characterized by its elegant and compact logic formalism, has traditionally operated in a collectionless manner, emphasizing the manipulation of symbolic knowledge without the need for amassing vast data collections. Conversely, Sub-symbolic AI, notably exemplified by the recent surge in Machine Learning (ML), has undergone a transformation fueled by the availability of extensive data repositories and the computational power to harness them, enabling the proliferation of deep neural architectures backed by statistical premises.

Intriguingly, as Machine Learning thrived on the influx of large-scale data collections (cfr. the historical example of ImageNet, Russakovsky et al. (2015)), it raised a fundamental question: Is the accumulation of copious data a necessity for AI systems, or is there an alternative path? Along these lines we might want to challenge the conventional wisdom that the path to AI progress inevitably leads to the storage and analysis of extensive data collections (Betti et al., 2021b; Gori and Melacci, 2023). Modern Machine Learning requires the storage of patterns, that apparently leads to the inevitable direction of accumulating big data collections. However, in the natural world, animals of all species navigate their environments and acquire knowledge without resorting to such massive data accumulation.

The concept of "Collectionless ML" (Gori and Melacci, 2023) identifies methods that operate on the data as they flow in and emphasizes the development of dynamic models capable of adapting over time. The process is not isolated; it hinges on environmental interactions, incorporating information from humans and potentially agent-to-agent communication. Moreover, it envisions AI systems that can be managed by edge computing devices, reducing the reliance on cloud computing and commercial entities.

There is the possibility that the need for vast data collections in machine learning stems from formulating the learning problem in a more complicated perspective than how natural learning occurs. When we consider the case of vision, there is a clear sequence in which images are processed, given by the natural evolution of

time and expressed by motion. Unlike machines, humans and animals do not learn by going through vast and randomly shuffled sets of images. Instead, they learn continuously, as they experience the world around them, without needing to store and label every piece of information they come across.

In this research, we argue that machines too can potentially adopt this natural way of learning. We believe that nature's way of learning has its foundations in how information is processed and used to adapt to environments. We propose a new approach for machines: let them learn continuously from their surroundings, just as living creatures do, without depending on large pre-existing datasets. This would involve, for example, machines processing visual information as it comes in and occasionally interacting with humans for guidance.

We acknowledge that proposing machines adopt a natural way of learning is an audacious objective, necessitating a profound paradigm shift. While the end goal may seem distant, this thesis seeks inspiration from this overarching vision. Through our research, we endeavor to take preliminary but significant strides in this direction, laying the groundwork for future exploration. In the following, we identify three specific limitations in the prevailing machine learning literature that have influenced our perspective. These shortcomings have inspired us to craft solutions that are in harmony with the philosophy outlined above.

A Traditional supervised learning in neural networks is predominantly viewed as a static process, a perspective that stands in stark contrast to the dynamism inherent in human cognitive development. Present-day neural networks are exposed to a diverse range of examples, spanning from simple to complex, right from inception. This approach is somewhat counter-intuitive when juxtaposed with human learning trajectories. Consider the pedagogical journey of a child: they aren't introduced to differential equations on their first day of school. Instead, they navigate through a meticulously structured path, starting with basic arithmetic and gradually advancing over the years.

Drawing a parallel in the realm of visual cognition, a newborn doesn't instantly possess the visual acuity of an adult (Braddick and Atkinson, 2011). On the contrary, infants need several months—and the progressive unlocking of different types of visual experiences (Smith and Slone, 2017)—to attain matured visual capabilities. It's plausible to surmise that this phased progression in visual acuity is a strategic biological adaptation, facilitating the step-by-step acquisition of strong visual competencies.

This leads to the following question:

*Can we design neural networks that emulate these natural, gradual learning processes? Is it feasible to enhance the performance and reliability of neural networks by institut-*

*ing a systematic learning schedule, mirroring the developmental stages observed in nature?*

- B The conventional understanding of "learning as a static process" in neural networks is grounded in situations where all training data is immediately available and sampled from a static distribution in an independent manner (i.i.d.). This paradigm holds up under such conditions, but becomes problematic in dynamic settings. Specifically, when data streams over time and is derived from evolving, non-stationary distributions, neural networks face challenges: a huge performance drop on old data is typically noticed when adapting parameters on novel data. This decline, known as catastrophic forgetting, is not a new challenge. It dates back to the initial phases of connectionist research and has remained an unresolved issue. A widely adopted mitigation strategy has been to buffer and periodically reuse a portion of old inputs during the learning process. While this can somewhat curb the effects of forgetting, continuous data retention ushers in a new set of complications, including storage challenges and privacy concerns.

This directs our attention to the following inquiry:

*Is it possible to deal with the continuous learning problem in a truly online and lifelong perspective without storing all the data?*

- C Historically, supervised task-specific models have been the landmark of progression in visual recognition systems. More recently, the landscape has been changed by the advent of unsupervised models for vision tasks. These models have been used as general-purpose feature extractors, bridging the performance gap with their supervised counterparts.

However, a critical bottleneck remains and it is given by their heavy reliance on vast collections of generic images, leading to what can be termed the 'accumulation problem'. This dependence on enormous datasets poses questions about efficiency and scalability. In this context, one wonders if there's a more organic source of data that these models can exploit. Video streams, teeming with temporal information, especially motion, present a compelling answer. Interestingly, Spelke (1990) showed that motion considerably augments the capability of biological perceptual systems to distinguish and segregate visual patterns into distinct entities.

This prompts us to ask:

*Given the rich, sequential data that can be extracted from video streams, can we leverage temporal evolution and motion to train our visual agents? Can we envision a paradigm where these agents learn predominantly by observing video streams, thereby significantly reducing the need for supervised guidance?*

## 1.2 Contributions

This thesis endeavors to answer the questions posed earlier in Sec. 1.1, grounding its investigations in a nature-inspired approach to prevalent machine learning challenges.

In the first part, we center our attention on ideas concerning the gradual fulfillment of learning objectives in vision tasks. We propose methods tailored for supervised learning in both stationary and non-stationary environments, providing a comprehensive overview of the varying challenges and solutions across these settings. The methods that we propose involve what we call *input tuning*, i.e., dynamic transformations of inputs over time, albeit evolved in diverse directions depending on the learning context.

- *Friendly Training*: We introduce a new method within the realm of curriculum learning. Applicable to any stationary supervised learning setting, this technique offers a more gradual human-like learning path for neural networks by transforming input samples, resulting in improved accuracy compared to standard training practices. *Based on (Marullo et al., 2021, 2022b).*
- *Continual Input Tuning*: We introduce a method for vision tasks within supervised continual learning contexts. By freezing a pretrained backbone and learning simple transformations in the input space, it provides a solution to mitigate forgetting without the need for data retention. *Based on (Marullo et al., 2023b).*

In the second part, our exploration pivots towards the autonomous cultivation of general-purpose visual abilities, conquered through the observation of unlabeled video streams. Our approach hinges on the harnessing of temporal information, linking consecutive frames via *motion*. This is why we start that discussion by investigating the feasibility of motion extraction in an online learning setting.

- *Optical Flow Estimation*: We undertake a comprehensive study of optical flow estimation in online continual learning. Our results suggest that the challenge of catastrophic forgetting is less pronounced for optical flow with respect to often investigated supervised tasks. We also offer straightforward strategies that enhance performance in common applications. *Based on (Marullo et al., 2022a).*
- *Self-Supervised Visual Feature Development*: We propose two methods aimed at developing robust visual features without supervision, solely by processing a video stream in real-time. The first approach is anchored on the principle of consistency across attention paths, while the second method more directly

exploits motion. Our results indicate that these features can be effectively used for downstream tasks like semantic segmentation. *Based on (Tiezzi et al., 2022; Marullo et al., 2023a).*

## 1.3 Structure of the thesis

This thesis is structured as follows:

- In Chapter 2, we briefly introduce the readers to the concepts of Curriculum Learning, Continual Learning, Visual Representation Learning.
- In Chapter 3, we describe a procedure, named Friendly Training, to transform a static learning problem into a sequence of small learning subtasks. We discuss two very different implementations, one is based on direct optimization, while the other features an auxiliary neural model. Both of them are based on transformations of the input, anticipating the notion of input tuning (Chapter 4) but in a different context. We compare the implementations and show the remarkable impact on several supervised learning benchmarks.
- In Chapter 4, we explore the idea of input tuning in the context of continual learning. We introduce a method to mitigate forgetting without buffering samples. We experimentally investigate two kinds of input transformation and we extend the method for large domain shifts. We outperform regularization-based competitors on several continual learning benchmarks.
- In Chapter 5, we investigate potential issues related to learning optical flow estimation in an online setting, exposed to non-stationary video streams. We present synthetic video streams of growing complexity and we discuss qualitative and quantitative results on a very-long realistic video stream. We evaluate forgetting issues and we suggest simple techniques to overcome common caveats.
- In Chapter 6, we present a technique for self-supervised learning based on the notion of human-like attention trajectory. Subsequently, we present a more involved technique, centered on the idea of joint learning of layered motion estimation and feature extraction. We evaluate the techniques on synthetic streams from 3D environments and real-world videos, showing that the methods are competitive with respect to large pretrained models.
- In Chapter 7, we reach conclusions regarding the work we have presented in this thesis, raising additional questions and proposing hypotheses for potential future research directions, along with a unifying perspective.

# Chapter 2

## Background: A developmental approach to learning

In this chapter we will provide the reader with further background, that we find useful to understand the spirit behind the contributions of the thesis and better contextualize the proposals with respect to the concerned research areas. We will begin with a developmental perspective on human learning and related machine learning research. Then we will briefly introduce to continual learning (also called life-long learning) in the machine learning paradigm. Finally, contrastive representation learning will be presented.

### 2.1 Human and Machine Developmental Learning

Learning can be defined as "adaptive, intelligent change in response to experience" (Smith and Slone, 2017) and is one of the most striking property of human intelligence. The human brain is a complex system that drastically changes in response to the activations evoked by sensory input over its developmental trajectory. Researchers in Artificial Intelligence always tried to emulate this fundamental capability in machine algorithms. One of the overlooked facets of human learning is the fact that the brain helps create the experiences from which we learn, for instance by directing the actions of the body. In so doing, the learner directly affects the properties of the raw material of learning with different effects at different maturational stages. This also means that learning in humans is not an uncontrolled process randomly consuming indiscriminate input stimuli until convergence, but rather seems to follow a meaningful schedule that leads to strong cognitive skills. We will use the adjective *developmental* to indicate the emphasis on the features of this non-stationary learning process over the temporal dimension.

The case of vision is particularly compelling because it can be readily explored in humans using cognitive psychology and neurophysiology tools, unveiling intrigu-

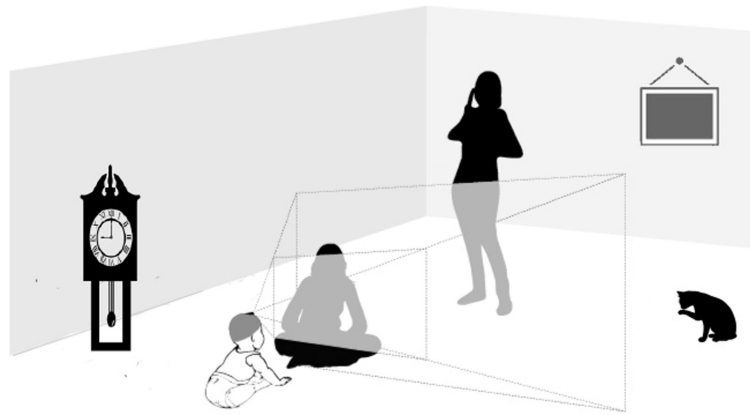


Figure 2.1: The selectivity of the ego-centric view in infants greatly shapes the content of visual data subject to the learning process. Unless they turn their head, elements like the cat or the clock remain hidden from view. The personal evolution in the way the observer stands, their movements, interests, and social interactions all influence the gradual acquisition of visual skills. Image courtesy of (Smith and Slone, 2017).

ing findings about natural learning environments. For instance, it has been known for a long time (Braddick and Atkinson, 2011) that the visual acuity of newborns drastically improves over the first six months of life. Apart from the curious post-natal evolution of the spatial resolution of the stimuli, the structure and patterns observed in such stimuli give rise to interesting observations. Such data are nothing like the training sets in cutting-edge computer vision since an individual’s perspective of the visual environment is highly selective (Fig. 2.1), influenced by a variety of factors (Smith et al., 2014). These factors—ranging from posture to personal interests—evolve over time, directing the individual towards specific types of visual experiences as they grow. Notably, in the initial two years of life, each sensory-motor milestone paves the way for novel visual experiences. Therefore, it can be sensibly stated that the human visual system does not just rely on batch learning of the world; it matures via a systematically structured visual experience curriculum, which is shaped by the infant’s sensory-motor progression. In contrast, typical machine vision (machine learning) does not experience anything akin to *learning that shifts the very nature of learning*. The notions that we summarized in this section have been highly inspirational for this thesis. The most directly related outcome is presented in Ch. 3, where we describe Friendly Training.

As a partial digression, the parallel with human development of visual skills let us draw more connections with the topics covered in this thesis. Indeed, acknowledging the multistage nature of visual learning allows to gain specific insights from the different stages. Visual learning in toddlers seems to exploit distinct characteristics of the sensory data that are useful to break down complexity. Firstly, toddlers,



when handling objects, produce visual scenes that are simpler than those observed in adults (Tamis-LeMonda and Masek, 2023). With their short arms, toddlers tend to lean in, examining objects up close, often resulting in a scene dominated by a single object. This setup inherently addresses several fundamental problems, including segmentation and competition. Secondly, they see multiple perspectives of the same item, linked by tactile connection (manipulation) and temporal proximity (motion). A logical conclusion would be that smart exploitation of the inherent structure of visual data predominantly addresses many of the complex challenges in visual object recognition. We will employ this intuition in Ch. 6, where we will devise a self-supervised development stage that makes use of attention and motion to conquer basic visual skills, postponing to a later maturational stage the ability of identifying entities with names.

## Curriculum Learning in Machine Learning

A lot of attention has been paid in the last decade, marked by the explosion of deep learning methodologies, to architectural aspects of learning. Comparably less attention has been paid to improving the training process and its material.

Nonetheless, the idea of training neural networks with a learning methodology that “gradually” changes the learning environment (which was proved to be realistic not only in humans but also in animals (Peterson, 2004)) traces back to almost three decades ago (Elman, 1993) and has been known for long as Curriculum Learning (CL) (Bengio et al., 2009) within Machine Learning. Taking inspiration from the common experience of learning in humans, CL aimed at designing an optimal learning plan in which the learning agent is exposed to simple, easily-discernible examples at first, and later to gradually harder examples, also progressively increasing the size of the training set.

Using the definition by Mitchell (1997), a model  $M$  is said to learn from experience  $E$  with respect to some class of tasks  $T$  and performance measure  $P$ , if its performance at tasks in  $T$ , as measured by  $P$ , improves with experience  $E$ . According to the formulation proposed by Bengio et al. (2009), curriculum learning corresponds with increasing the complexity of the data samples over the training process. With respect to the aforementioned definition, this means that the complexity of  $E$  grows over time. However, curriculum techniques have been investigated on other components of the learning interplay, for instance by increasing the model capacity  $M$  (Sinha et al., 2020; Morerio et al., 2017). In any case, Bengio et al. (2009) interestingly notices that curriculum learning can be generally interpreted as instances of continuation methods (Allgower and Georg, 2003), well known in non-convex optimization. Continuation methods start with a relatively simple and smooth objective function that is easy to optimize. Over time, this objective function is gradually

transformed into more complex and less smooth versions until it finally matches the original, non-convex objective function.

Coming to training in machine learning, when we use straightforward data samples at the beginning we naturally expect our model to quickly achieve good performance levels. This is justified by the fact that the objective function, associated to measure  $P$ , is smoother and easier to work with (Bengio et al., 2009). As we introduce more challenging data samples, the objective function becomes more difficult to deal with. A similar argument applies for curriculum applied to our model  $M$  or the tasks  $T$ . Drawing a connection between curriculum learning and continuation methods helps us see that the ultimate goal is to make the optimization process smoother in the initial training phases.

Wu et al. (2021) focused on curriculum methods applied to experience  $E$ , identifying three main components. The crucial one is the scoring function  $s(x)$ , mapping every example to a numerical score reflecting the concept of difficulty. The pacing function, denoted as  $g(t)$  governs the size of the training dataset used at each training step. Generally, an ordering scheme can be specified: curriculum (ordering examples from lowest score to highest score), anti-curriculum (ordering examples from highest score to lowest score), or random (standard training).

In the broader realm of Curriculum Learning, we mention Self-Paced Learning (SPL), which emphasizes the fact that the order of examples is not initially known but is dynamically computed based on the model's evolving performance. For instance, in the work by Kumar et al. (2010), they leverage the likelihood of predictions to determine the ranking of samples.

---

Along this lines, Smith and Slone (2017) compared ideas behind curriculum learning and current notions in developmental learning. They criticize plain curriculum learning because visual data in infants are not just ordered over developmental time, but they are also dynamically structured by the current state of the learner, adapting as the learner's internal system evolves. From what we understand of human cognitive development, the information presented at any given moment is likely the most suitable for the current stage of learning, ensuring that the right information is delivered precisely when needed. This is exactly the goal that we seek to achieve in Ch. 3 with Friendly Training.

## 2.2 Continual and Lifelong Machine Learning

Humans continually adapt their knowledge, learning concepts sequentially. Although they might occasionally revisit old concepts, they usually retain this knowledge without needing constant reinforcement. Conversely, artificial neural networks

tend to forget previously learned concepts as they acquire new ones (French, 1999). It looks like the impressive accomplishments of Machine Learning stem from static models that do not adapt over time. To accommodate new data, retraining is typically necessary, raising theoretical and practical issues (including storage and privacy concerns).

Historically, artificial neural network research has emphasized static tasks, shuffling data to achieve i.i.d. conditions, and improving performance by revisiting training data. Continual Learning, also known as lifelong (Chen and Liu, 2018), sequential (McCloskey and Cohen, 1989), or incremental learning (Gepperth and Karaoguz, 2016), addresses the challenge of learning from an endless data stream. The primary goal is to extend knowledge over time without significant performance degradation on older tasks as new tasks emerge. This issue stems from the broader stability-plasticity dilemma in neural networks (Grossberg, 2012), balancing the integration of new knowledge and the retention of existing knowledge.

Traditionally, the ambition of continual learning has been initially developed under the term *lifelong learning* (LL), introduced by Thrun and Mitchell (1995). Initial works were in field of information systems, since in that case continual learning was very intuitively implemented with mechanisms to grow the knowledge base over time. NELL (Never-Ending Language Learner) is a prominent example of lifelong learning in such field (Carlson et al., 2010; Mitchell et al., 2015). NELL has come to life with a small ontology plus a collection of 500 million web pages. With access to the remainder of the web through search engines, it has run 24 hours per day from 2010 to 2018, extracting new instances of categories and relations. Simultaneously, NELL performed retraining of belief extraction methods, using the growing knowledge base as a self-supervised collection of training examples. Moreover, periodic review sessions have been organized, with human operators providing corrections and introducing new tasks.

In recent years, the landscape of lifelong learning has undergone a significant transformation, largely attributed to the widespread adoption of deep learning. This shift has given rise to a thriving community dedicated to similar ambitions, under the umbrella of *continual learning*. Within the realm of deep learning, researchers have delved into the complex challenge of continually acquiring expertise in a sequence of tasks (Parisi et al., 2019).

The primary motivation behind continual learning in the context of deep learning stems from the need to address the issue of catastrophic forgetting that can occur when sequentially acquiring knowledge in a series of tasks. The central focus has been on the incremental acquisition of each new task within the same neural network framework, all while ensuring that the neural network retains the previously acquired knowledge for past tasks. This approach stands in contrast to traditional lifelong learning literature, which often emphasized explicit techniques for leverag-

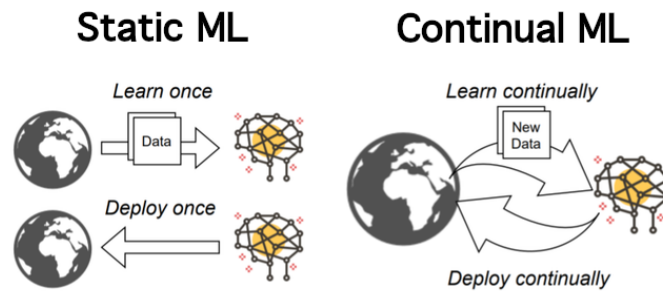


Figure 2.2: Traditional machine learning models are trained once using available data and then remain static when deployed. This rigidity is problematic in our dynamic world, especially since such static models can't leverage continuous data streams (e.g., IoT). Continual Learning designs techniques for models that adapt to data, continuously and flexibly. Image courtesy of Leuven.AI.

ing prior knowledge to aid in the acquisition of new skills.

Literature about Continual Learning has rapidly grown with a multitude of different approach and little consensus on settings, learning protocols and methodologies, leading to the need of creating taxonomies. It is important to emphasize that while the majority of studies focused on supervised learning, unsupervised settings received comparatively minimal attention (Madaan et al., 2022; Rao et al., 2019; Tiezzi et al., 2020; Betti et al., 2022a). One of the most popular categorization is the tree structure proposed by De Lange et al. (2021). We report here the main categories, since we will use these concepts to contextualize our proposals, as it will be more clear in Chapters 4, 6.

- **Replay methods.** This line of research involves the preservation of samples with buffers and specifically designed policies. Rehearsal methods (Rebuffi et al., 2017; Rolnick et al., 2019) involve explicit retraining on a limited subset of the stored samples while concurrently training on samples from new tasks. Inspired by knowledge distillation, Buzzega et al. (2020) proposed to store in the buffer not only the ground-truth labels, but also the logits acquired during the optimization trajectory. Noteworthy, traditional rehearsal methods are of course bounded by joint training (on previous and current tasks). As a consequence, more sophisticated methods emerged (Lopez-Paz and Ranzato, 2017; Chaudhry et al., 2019), based on constrained optimization, which should allow for greater flexibility in the transfer of knowledge. The fundamental concept is to ensure that updates made for the new task do not adversely affect (i.e., reduce the performance of) previously acquired task knowledge. While replay methods are the most popular category in CL, in Chapters 4, 6 we consider

scenarios where large availability of past samples is out of the picture, and we propose methods where replay is absent or marginal.

- **Parameter isolation methods.** In this approach, distinct sets of model parameters are allocated to individual tasks to prevent any potential interference or forgetting. In cases where architectural constraints are not a concern, new branches can be introduced for each new task while keeping the parameters of previous tasks frozen (Rusu et al., 2022; Masana et al., 2022), or even by dedicating a separate model copy to each task (Aljundi et al., 2017). It is important to note that these approaches typically require to know or estimate task-specific information, often referred to as a "task oracle," to activate the corresponding branches during prediction. In Chapter 4, we will present an heuristic-based method of this category, in the context of continually tuning pretrained backbones.
- **Regularization methods.** This line of works is united by the idea of adding regularization term within the loss function, with the broad aim of consolidating prior knowledge when learning from new data. Methods of this category have typically lower storage requirements and emphasize privacy preservation. They can be categorized into two groups: data-focused and prior-focused.

Data-focused methods primarily rely on knowledge distillation from a previously trained model (on a prior task) to the model undergoing training on new data. This concept, popularized in (Li and Hoiem, 2017), combats forgetting by employing the previous model's output as soft labels for earlier tasks. In contrast, prior-focused methods concentrate on estimating a distribution over the model parameters, which serves as a prior when learning from new data. Typically, the importance of all parameters is assessed and changes to important ones are penalized (Kirkpatrick et al., 2017; Zenke et al., 2017). In Chapter 4 we will adopt competitors of this category for experimental comparison. In Chapter 6 we will propose methods for online unsupervised learning of visual features. While they are not properly represented in the dichotomy between data-focused and prior-focused, we argue that they still fall under the umbrella of regularization methods for continual learning, although in a very specific context, as it will be clear later.

### Alternative taxonomies

Apart from this traditional categorization of continual learning, other taxonomies (Wang et al., 2023; van de Ven et al., 2022) have been proposed, taking into account more recent efforts in the research community. We will briefly mention a few directions, since they are useful to contextualize the works discussed in this thesis.

**Representation-based methods.** Some approaches harness representations for continual learning by integrating self-supervised learning (SSL) (Madaan et al., 2022) and large-scale pre-training (Mehta et al., 2023). These two strategies often intersect, given that modern pretraining of large models substantially identifies with SSL on abundant, unlabeled data. An interesting niche combines self-supervised learning with traditional supervised loss, often employing the complementary learning system paradigm (Pham et al., 2021). Techniques for large-scale pre-training focus on adapting a fixed backbone, involving extra neural modules (Ermis et al., 2022) or enriched input prompts (Wang et al., 2022b,a). We exploit SSL in Chapter 6.

**Template-based methods.** In the context of classification tasks, prevalent in continual learning, template-based classification (Rebuffi et al., 2017; Hayes and Kanan, 2020) is common. Each class is represented by a template (as in Ch. 6), typically defined as the mean vector in an embedding space. At inference time, samples are assigned to the class with the closest prototype. In class-incremental learning, where the embedding network remains fixed, a single prototype per class suffices. However, when the embedding network evolves, methods store examples for each class (De Lange and Tuytelaars, 2021), ensuring prototypes adapt without drifting.

**Task-free methods.** The vast majority of methods for continual deep learning work within a task-based sequential learning framework. Boundaries between tasks are always available during training, while sometimes task identity is obscured at inference time. However, this task-based structure is seldom seen in real-world applications. On the contrary, Aljundi et al. (2019) praiseworthily introduced the case of systems continuously learning in a streaming/online manner, where data distributions evolve over time without distinct task demarcations (as in Chapters 5, 6). While this direction is extremely interesting, it did not receive much attention apart from a few exceptions (Ye and Bors, 2022; Lässig et al., 2023). Sometimes these methods involve heuristics for detection of data shifts, hence they can be described as applying generic CL methods on *pseudotasks*.

---

Undoubtedly, the principal trajectory in Continual Learning maintains a strong connection with the traditional Machine Learning paradigm, underpinned by statistical principles related to supervised modelling of some training distribution. For instance, these approaches aim to adapt incrementally to new tasks while still learning from shuffled and isolated samples. A small minority of works (Mai et al., 2022) correctly highlights the fact that most of the proposed algorithms are tested with models being trained in an offline manner, with repeated shuffle for multiple epochs on the individual tasks. However, this setting requires storing all data from the current task for training, which may not be feasible due to privacy issues or resource

limitations. Following this remark, in Chapters 5, 6 we will explicitly consider the issue when presenting our proposals in the continual learning domain.

This traditional connection represents a notable limitation in the field, as it tends to overlook the critical imperative of designing continual learning solutions that can effectively operate in dynamic and unstructured environments. In the field of visual recognition, the prevalent approach tends to overlook the temporal dimension present in human visual experiences. Instead, it supposedly simplifies the problem by focusing on static images, neglecting the patterns and regularities found in temporal data sequences. This choice may introduce unnecessary complexity compared to the more natural learning process seen in biological systems. We will deepen this discussion in Chapter 6.

## 2.3 Visual Representation Learning

From an intuitive standpoint, it's impractical to equip an artificial agent with all the prerequisite knowledge needed to function efficiently in real-world scenarios (Thrun and Mitchell, 1995). Therefore, when viewed from a developmental perspective, artificial agents should possess learning capabilities that allow them to interpret environmental cues, at least to some extent, autonomously. On the other hand, we know that deep convolutional neural networks (Krizhevsky et al., 2012) have revolutionized visual tasks thanks to effective exploitation of large annotated visual collections. Summarizing a large part of computer vision advancements of the last two decades, standardized image datasets have driven progress together with data augmentation, optimization techniques, and improved training methods.

The substantial reliance on labelled visual collections rises practical issues, apart from methodological ones. For instance, not all problems offer ample labeled training data, and this is demonstrated by the rise of transfer learning (Tan et al., 2018) as a popular solution. While effective, exploiting features extracted from models trained on larger collections comes with limitations and unexpected brittleness (Jain et al., 2023).

Historically, limitations of transfer learning and plateauing of supervised learning models on common visual benchmarks boosted the attention for unsupervised learning and, more precisely, self-supervised learning. The main idea behind self-supervised learning is rather simple and consists in designing learning environments where the supervision signal is already available and does not need to be collected with human intervention. A basic example drawn from language is the task of learning to complete a sentence in a plausible way, exploiting text collections from the Web.

Efforts in self-supervised learning for natural language processing (NLP) overcame the supervised counterpart earlier (Devlin et al., 2019), due to abundant un-

labeled text data, other than a different problem structure that is easier to frame in the SSL paradigm. However, self-supervised learning in computer vision emerged (Oord et al., 2018; He et al., 2020).

Self-supervised learning methods can be categorized into two main types: generative and discriminative. Generative self-supervision, including auto-encoders (Kingma and Welling, 2013) and generative adversarial networks (Goodfellow et al., 2020), focuses on constructing distributions over data, typically in pixel space. However, it has faced criticism for being computationally expensive, and possibly superfluous (Chen et al., 2020a) for the purpose of representation learning.

In this thesis (see Chapter 6), we will use notions from discriminative self-supervision, which aims to learn effective data representations for specific pretext tasks without relying on human annotations. This approach loosely resembles supervised learning, as it often involves an objective function that evaluates the discriminative capabilities of learned representations.

## Popular concepts in Contrastive Representation Learning

In traditional supervised learning, the training process involves simultaneously optimizing both the feature extractor (e.g., convolutional layers) and the predictor (e.g., linear layers responsible for mapping features to class labels). However, self-supervised learning (SSL) sharply separates the two phases in time. Once SSL training concludes, the trained backbone is kept and the additional module, responsible for the accomplishment of the downstream task, is trained.

Coming to actual SSL training, such methods acquire meaningful representations through pretext auxiliary tasks. Within auxiliary pretext methods, the model autonomously garners supervisory signals directly from the data, sidestepping the need for manual annotation.

Creating a suitable pretext task necessitates domain-specific expertise, constituting a pivotal aspect of SSL. The beauty of this approach lies in its versatility, as pretext tasks can be tailored to diverse data types, encompassing audio, text, images, and videos (Jaiswal et al., 2021). Historically, one of first pretext tasks to be investigated in the visual domain was image colorization (Charpiat et al., 2008), even before the advent of deep networks. Clearly, turning a grayscale image into a plausible colorized one requires understanding what is depicted, and this involves obtaining meaningful representations of the input. Once we have a collection of color images, we can easily setup the learning task without any human-provided annotation. Similarly, the spectrum of pretext tasks includes predicting missing pixel values (Pathak et al., 2016), jigsaw puzzles (Noroozi and Favaro, 2016), among others. However, these handcrafted methodologies have been generally overcome by an even simpler idea, known as *instance discrimination*. Such technique involves the recognition of variously augmented views (referred to as instances) originating from a single



image. The goal is to correctly identify these views as originating from the same image while distinguishing them from any views of a different image source (Wu et al., 2018)—see Fig. 2.3. This approach differs from the pretext tasks discussed

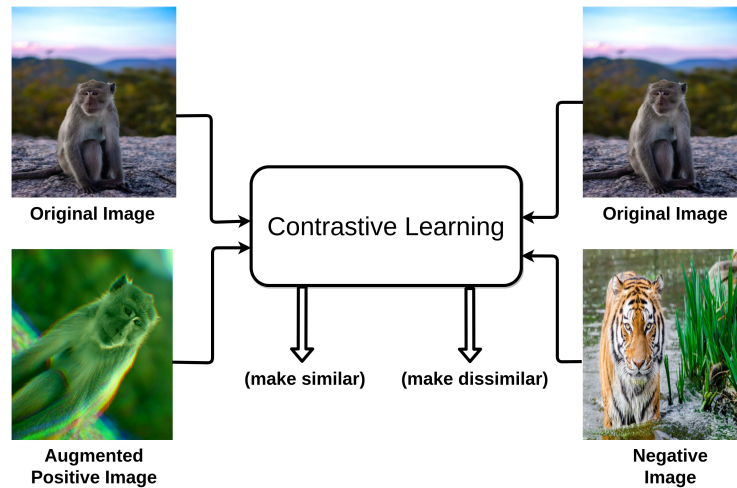


Figure 2.3: At the heart of the instance discrimination task in the contrastive learning paradigm: draw representations of original and augmented images closer in the latent space, while pushing the original away from negative images. Image courtesy of (Jaiswal et al., 2021).

earlier, which indirectly achieve representation learning while optimizing for other objectives. Differently, instance discrimination directly optimizes for representation learning by aligning or contrasting representations. Instance discrimination has been so influential in self-supervised learning that the resulting family of methods, contrastive methods, are often considered (Ozbulak et al., 2023) the most representative ones in the self-supervised field.

### Positive and negative samples: contrastive losses

In this context, images that are contrasted against the similar ones are termed *negative* samples. The core concept underpinning representation matching between similar images (positive samples) and differentiation from dissimilar ones (negative samples) is to facilitate the training of deep neural networks to learn representations that remain consistent even in the face of common image transformations. In fact, most of these transformations do not alter the underlying visual semantics (Misra and van der Maaten, 2020) and they can be considered a reasonable approximation of visual variability observed in the real world.

In popular contrastive self-supervised learning frameworks, a variety of loss functions have been proposed for effective training. While these functions are often framework-specific, we will briefly review some of the most commonly used terms, which will also be employed in this thesis.

Among similarity measures, **cosine similarity** has a long history. This metric quantifies the similarity between two non-zero vectors in a high-dimensional space, calculated as the dot product between  $\ell_2$ -normalized vectors  $r_1$  and  $r_2$ .

$$\text{sim}(r_1, r_2) = \frac{r_1 \cdot r_2}{\|r_1\| \|r_2\|}$$

Cosine similarity is frequently used to quantify representation similarity, often in conjunction with **noise-contrastive estimation** (NCE) in contrastive learning.

Among contrastive losses, **InfoNCE** (Oord et al., 2018) has been a crucial loss function in image-based SSL, derived from NCE (Gutmann and Hyvärinen, 2010). The main idea is to use categorical cross-entropy loss to identify the positive sample amongst a set of unrelated noise samples. Using the formulation popularized by Chen et al. (2020a), InfoNCE – also termed by the authors as "normalized temperature-scaled cross entropy loss" – can be implemented at batch level using two augmentation functions  $\mathcal{T}_1, \mathcal{T}_2$ . Such functions are used to produce two augmented views for each sample in a batch  $x$ , so that

$$x = (\mathcal{T}_1(x_1), \mathcal{T}_2(x_2), \dots, \mathcal{T}_1(x_n), \mathcal{T}_2(x_n))$$

resulting in a batch of  $2n$  samples. If  $(i, j)$  is a positive pair, i.e., the two augmented views of the same original sample, InfoNCE is computed as

$$\mathcal{L}_{\text{InfoNCE}}(x_{\{i,j\}}) = -\log \frac{\exp(\text{sim}(r_i, r_j))}{\sum_{k=0}^{2n} \mathbf{1}_{k \neq i} \exp(\text{sim}(r_i, r_k))}$$

with  $r_z$  the features extracted by the neural model with the input  $x_z$ . In order to compute the loss value for the batch, the computation is repeated for each positive pair.

### Siamese networks and learning dynamics

Siamese networks, a computational model consisting of two identical neural networks sharing the same weights, have been introduced way earlier than modern contrastive learning. Initially proposed for pattern verification and other specific tasks (Bromley et al., 1993), they are widely used in self-supervised learning (SSL) methods, where they are employed to ensure consistency between representations. In this context, the Siamese network is just an abstraction to represent the fact that two inputs, derived from some original sample, are processed by the same neural network. Then the loss is computed and then backpropagation is performed through both the "virtual branches". Very often, however, the two branches are not identical, so that the term *dual-backbone* architecture should be preferred, meaning that the two branches are actual physical branches with non-overlapping set of parameters. The term "stop-grad" denotes the practice of halting the gradient flow from

one branch of a dual-backbone network while allowing it to update the weights of the other branch, introducing a strong asymmetry in the learning dynamics (Chen and He, 2021).

Now, consider a Siamese-like dual-backbone network where one branch is referred to as the teacher, and the other as the student. Unlike the traditional Siamese architecture, these models do not share weights. In this setup, delayed-weight updates involve the idea of backpropagating errors through only one branch and updating the trainable parameters of the other branch using a specifically predetermined rule. Common implementations are Mean Teacher (Tarvainen and Valpola, 2017) and momentum encoding (He et al., 2020).

---

We have introduced a few popular concepts in modern Self-Supervised Learning, although used in a quite different spirit with respect to what we do in this thesis. In fact, we elect motion as the primary source of information for visual learning. In the design of self-supervised online-learning visual agents in Chapter 6, we will draw inspiration from dual-backbone networks, as well as stop-grad asymmetry and delayed weight update mechanisms. Moreover, we will design an InfoNCE-like loss that does not compare image global representations but rather pixel-wise local representations, according to a motion-based criterion.

# **Part I**

## **Progressively learning in the input space**

## Progressively learning in the input space

In this part of the thesis, we delve into novel training techniques for neural networks that involve dynamic transformations of inputs over time. Throughout this exploration, we will use the term *input tuning* to refer to such an idea. The investigation unfolds in two chapters, each addressing a distinct facet of the learning landscape. As we discussed in Introduction and in Chapter 2, our exploration of shaping data in the input space is also partially inspired by the intriguing process observed in infants. At birth, visual information in newborns is throttled, and its structure follows a clear progression that results in infants gradually acquiring visual skills over time.

In Chapter 3, we draw inspiration from Curriculum Learning, navigating the motivations behind this approach to craft smooth learning trajectories. Here, the core idea lies in breaking down a single, complex learning problem into a series of smaller subtasks, each incrementally more challenging than the last. We achieve this by implementing a learning schedule through input transformations, tailoring the cognitive complexity to match the learner’s current skill level. This methodology not only enhances the model’s performance but also fosters a more nuanced understanding of the learning process.

Chapter 4 addresses a distinct but closely related challenge. In fact, both Curriculum and Continual Learning fundamentally consider a learning process that unfolds over time, where each stage significantly impacts the final outcome of the entire process. Here, we consider a scenario where the learner, already equipped with certain cognitive skills, encounters a non-stationary environment presenting various tasks of differing complexities and spanning diverse regions of semantic and perceptual spaces. Unlike the controlled subtasks of Ch. 3, these tasks are externally given and lack a cognitively meaningful order. This dynamic setting raises concerns about catastrophic forgetting issues if learning proceeds naively. To mitigate this, we propose a novel training method. In this approach, we freeze the learner’s core while delegating environment adaptation to learnable transformations applied in the input space. This technique, inspired by the dynamic input manipulations of Ch. 3, provides an effective shield against forgetting.

In this part of thesis, we delve into a novel approach within the domain of neural network training procedures. Unlike conventional methods that primarily concentrate on enhancing learner models through ad-hoc components and specialized loss

functions, we advocate for a shift in focus. We redirect the designer's attention from the learner model to the input space. Central to our investigation is the necessity for models to evolve and adapt over time.

Our work emphasizes the pivotal role of preprocessing samples through learnable transformations, thereby enabling neural networks to learn gradually in evolving, non-stationary environments. We argue that shaping data in the input space is a promising direction towards increasing efficiency, resilience, and accuracy of machine learning systems in dynamic, real-world scenarios.

## Chapter 3

# Input tuning for Friendly Training

In the last decade, motivated by the success of Deep Learning, the scientific community proposed several approaches to make the learning procedure of Neural Networks more effective. When focusing on the way in which the training data are provided to the learning machine, we can distinguish between the classic random sampling of stochastic gradient-based optimization and more involved techniques that devise curricula to organize data, and progressively increase the complexity of the training set. In this chapter, we propose a novel training methodology named Friendly Training that, differently from the aforementioned approaches, involves altering the training examples in order to help the model to better fulfil its learning criterion. The model is allowed to “simplify” those examples that are too hard at a certain stage of the training procedure, creating an instance of what we call *input tuning* for curriculum learning. The data transformation is controlled by a developmental plan that progressively reduces its impact during training, until it completely vanishes. In a sense, this is the opposite of what is commonly done when incorporating adversarial examples in training data, i.e., Adversarial Training. In order to actually produce the transformed examples, two procedures are proposed, one is directly based on an optimization scheme, while the other involves the usage of an auxiliary neural model. Although our primary concern is for visual data, experiments on multiple datasets are provided, including some non-visual tasks. We show that Friendly Training yields improvements with respect to informed data sub-selection routines and random selection, especially in deep convolutional architectures aided by the auxiliary model. Results suggest that adapting the input data is a feasible way to stabilize learning and improve the generalization skills of the network.

## 3.1 Introduction

In the last decade, the scientific research in neural networks studied different aspects of the training procedure, leading to deep neural models of significantly increased quality (Ioffe and Szegedy, 2015; Kingma and Ba, 2015; Srivastava et al., 2014; Bengio et al., 2009; Li and Gong, 2017; Zhang et al., 2020a). Amongst a large variety of approaches, this chapter considers those that are mostly oriented in performing specific actions on the available training data in order to improve the quality of the trained neural classifier. For example, Curriculum Learning (CL) pursues the idea of presenting the training data in a more efficient manner (Bengio et al., 2009; Wu et al., 2021; Sinha et al., 2020), exposing the network to simple, easily-discernible examples at first, and to gradually harder examples later, progressively increasing the size of the training set (see Sec. 2.1 for an introduction). Self-Paced Learning (SPL) (Kumar et al., 2010) is another related research area, in which some examples are either excluded from the training set or their impact in the risk function is downplayed if some conditions are met (Li and Gong, 2017).

A common property of CL and SPL is that they essentially sub-select or re-order the training examples, without altering the material of the learning process. On the contrary, in this section we will introduce Friendly Training (FT), a novel approach characterized by transformations on the inputs during the tuning of the learner's parameters, facilitating the early fulfilment of the learning criterion. Basically, data are modified to better accommodate the development of the classifier. Such transformations (also referred to as "simplifications") are controlled and embedded into a precise developmental plan in which the training procedure is progressively constrained to reduce their extent, until data are left in their original version. A key property of FT is that data are altered according to the state of the classifier at the considered stage of the training procedure. This concept is coherent with the developmental mechanisms of human brain (Sec. 2.1), where the visual input at any moment depends on the current state of the learner, providing just the right information at the right time. We will discuss a first implementation, Optimization-based Friendly Training (OFT), where each example is perturbed by a specific offset, obtained by an inner iterative optimization procedure that is started from scratch for each input.

In this chapter we will also extend the simple idea of FT in another direction, introducing an auxiliary neural model. Therefore, we will refer to this second approach with the acronym NFT (Neural Friendly Training). The intuition is that the data simplification process of FT might include regularities that are shared among different training examples, and that there is an intrinsic coherence in the way data are altered in consecutive training iterations, i.e., similar simplifications might be fine in nearby stages of the training procedure. These considerations are not ex-



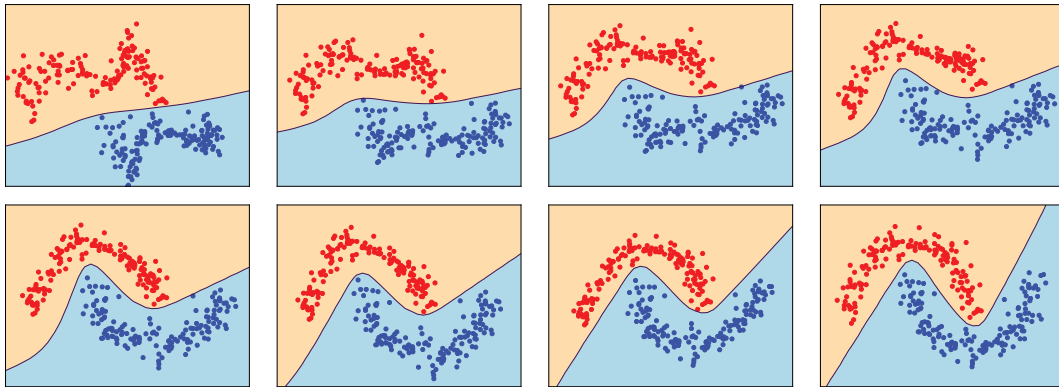


Figure 3.1: Left-to-right, top-to-bottom: evolution of the decision boundary developed by a single hidden layer classifier (5 neurons) in the 2-moon dataset, in Neural Friendly Training. Each plot is about a different training iteration ( $\gamma$ ); in the last plot data are not transformed anymore.

exploited by OFT, which applies an independent perturbation to each example, estimated from scratch at each training step. We propose to introduce an auxiliary multi-layer network, that is responsible of altering data belonging to the input space of the classifier. The auxiliary network is trained jointly with the neural classifier, and it learns how to transform the data to improve the learning process of the classifier itself. The weights of the auxiliary net represent the state of the alteration model, that is progressively updated by the training procedure, thus letting the model evolve as long as time passes. From an architectural perspective, the auxiliary network extends the classifier by adding a new set of initial layers, thus increasing the “depth” of the model. The effect of the auxiliary network is progressively reduced until the end of training, when it is fully dropped and the classifier is deployed for applications. Fig. 3.1 illustrates the behaviour of NFT in a toy 2D classification problem.

The contributions of this chapter are: (1) we propose a novel training strategy, named Friendly Training, that allows the machine to partially simplify the data by automatically determining how to alter it; (2) we propose a first implementation, Optimization-based Friendly Training (OFT), together with a developmental plan that creates a smooth transition from simplified data to the original one; (3) we propose another implementation (Neural Friendly Training, NFT) that allows the machine to simplify the training data by means of an auxiliary network that progressively fades out; (4) we experimentally compare OFT and NFT, using convolutional and fully-connected neural architectures. Our results confirm that NFT outperforms OFT, proving that NFT is a feasible and effective way to improve the generalization skills of the network and to efficiently deal with noisy training data.

## 3.2 Relation with existing works

### Curriculum Learning (CL)

The idea of training neural networks with a learning methodology that “gradually” changes the learning environment traces back to almost three decades ago (see Sec. 2.1). The concrete effect of CL<sup>1</sup> has been recently surveyed by Wu et al. (2021), showing that it is particularly evident in noisy learning settings or large-data regimes with limited training time. Even though CL has not been widely adopted in the machine learning community, there is strong evidence about the dramatic impact of the learning environment in the early stages of deep network training, affecting the network behavior at steady state (Jastrzebski et al., 2019). For instance, Achille et al. (2018) showed that inflicting visual impairments to deep CNNs in the first epochs leads to lesser visual skills despite conceding recovery time.

The purpose of Friendly Training, on the opposite, is to downplay difficult, confusing and outlier examples at the beginning, while letting them contribute to the generalization capability when the learner has already acquired basic skills. With respect to CL, Friendly Training does not alter the size of the dataset as a function of the number of training iterations, nor the relative ordering in which examples are presented to the network. Moreover it does not assume any predefined complexity criteria. Conversely, Friendly Training alters every single example in an adaptive way, and afterwards the examples are used to update the network weights. In so doing, the network is exposed, at every training step, to a data population with quite large variability, although their distribution has been slightly adapted, in order to decrease the occurrence of abrupt weight changes.

### Self-Paced Learning

Self-Paced Learning (SPL) (Kumar et al., 2010) is a technique inspired by CL, originally designed for Structural Support Vector Machines (SSVMs) with latent variables. Self-paced stands for the fact that the curriculum is determined by the pupil’s abilities (the classifier’s behaviour) rather than by a teacher’s plan. Since this direction proved to be effective when compared with standard algorithms (latent variable models correspond to hard optimization problems), researchers adapted this formulation to CNNs (Li and Gong, 2017). The basic idea consists of searching for suitable example-specific weighting coefficients in the loss computation. In contrast to this approach, what we propose is not about weighting the importance of training examples, but rather temporarily altering them. However, we do embrace the idea that the state of the classifier is what can be used to determine how to deal with a

---

<sup>1</sup>Notice that the acronym *CL* is used in this chapter for *Curriculum Learning*, while in the rest of the thesis it is used for *Continual Learning*.

particular input/target pair, gradually exposing the learner to more and more difficult examples, with an explicit temporal dynamics.

## Friendly Adversarial Training

Another technique that is somehow related to Friendly Training is the so-called Adversarial Training (AT) (Goodfellow et al., 2015; Madry et al., 2018), which may be seen as the inverse learning technique of FT. Developed as an empirical defense strategy for adversarial attacks, which seriously affect neural networks operating on high-dimensional spaces (Goodfellow et al., 2015), AT incorporates adversarial data into the training process. Most AT techniques rely on a minimax optimization problem (Madry et al., 2018), since the goal is to generate adversarial examples that strongly fool the classifier. Each generated example is an artificial element, lying very close to an original training datapoint of a certain class, but that is misclassified (i.e., classified incoherently with respect to the label attached to such original datapoint). Interestingly, in AT the system basically alters examples as in Eq. 3.2, even though the perturbation is computed with a different criterion and typically with no temporal dynamics.

Friendly Adversarial Training (FAT) (Zhang et al., 2020a) builds up on the ideas of both CL and AT. Tsipras et al. (2019) noticed that the adversarial formulation sometimes hurts generalization capabilities (Raghunathan et al., 2019). The FAT strategy provides a more gentle learning problem, in which the generation of adversarial data is early-stopped as soon as the datapoint is misclassified. The resulting learning dynamics is such that, as learning progresses (together with accuracy and robustness), more and more iterations are needed to generate (harder) adversarial data. Our OFT algorithm (see Sec. 3.3) shares many intuitions with the FAT algorithm (e.g., early-stopping), even though the direction of the iterative process which generates input data has a different goal. Moreover, FAT does not include explicit temporal dynamics.

## Auxiliary neural models to alter data samples

Neural models to alter data samples have been proficiently exploited by the Adversarial Machine Learning community (Qiu et al., 2020; Xiao et al., 2018) with the goal of fooling a classifier. Once trained, such generators can be used to generate adversarial perturbations efficiently, so as to potentially accelerate empirical defenses like adversarial training. When considering how to improve a classifier exploiting another network, a connection can be traced with Knowledge Distillation (KD) (Hinton et al., 2014; Phuong and Lampert, 2019), although in KD the main network is supplied with output probability distributions obtained from a pretrained large model. On the other hand, the auxiliary network of NFT (Sec. 3.3) learns to trans-

form the input data, closer to what is done by Spatial Transformer Networks (Jaderberg et al., 2015). Authors decided to focus on image data only and well-defined geometric transformations, proposing a differentiable learnable module that can be inserted into existing convolutional architectures. The main idea is to use a *localization network* to estimate the parameters of a spatial transformation (chosen from a pre-defined family) that helps the network to achieve correct predictions, obtaining some robustness to scale, rotation and distortion. While it clearly shows the concept of an auxiliary network that *simplifies* input data, Spatial Transformers deal only with predefined geometric transformation without a developmental perspective.

### 3.3 Friendly Training: implementations

We consider a generic classification problem in which we are given a training set  $\mathcal{X}$  composed of  $n$  supervised pairs,  $\mathcal{X} = \{(x_k, y_k), k = 1, \dots, n\}$ , being  $x_k \in \mathbb{R}^d$  a training example labeled with  $y_k$ .<sup>2</sup> Given some input data  $x$ , we denote with  $f(x, w)$  the function computed by a neural network-based classifier with all its weights and biases stored into vector  $w$ . When optimizing the model exploiting a mini-batch based stochastic gradient descent procedure, at each step of the training routine the following empirical risk  $L$  measures the mismatch between predictions and the ground truths,

$$L(\mathcal{B}, w) = \frac{1}{|\mathcal{B}|} \sum_{i=1}^{|\mathcal{B}|} \ell(f(x_i, w), y_i), \quad (3.1)$$

where  $\mathcal{B} \subset \mathcal{X}$  is a mini-batch of data of size  $|\mathcal{B}| \geq 1$ ,  $(x_i, y_i) \in \mathcal{B}$ , and  $\ell$  is the loss function. Notice that, while we are aggregating the contributes of  $\ell$  by averaging over the mini-batch data, every other normalization is fully compatible with what we propose. In the most common case of stochastic gradient optimization, a set of non-overlapping mini-batches is randomly sampled at each training epoch, in order to cover the whole set  $\mathcal{X}$ . We will refer to what we described so far as Classic Training (CT).

#### 3.3.1 OFT: direct-optimization model

CT provides data to the machine independently on the state of the network and on the information carried by the examples in each  $\mathcal{B}$ . However, data in  $\mathcal{X}$  might include heterogeneous examples with different properties. For instance, their distribution could be multi-modal, it might include outliers or it could span over several disjoint manifolds, and so on and so forth. Existing results in the context of CL (Bengio et al., 2009; Wu et al., 2021) and SPL (Li and Gong, 2017) (Section 4.1) show that

<sup>2</sup>We consider the case of classification mostly for the sake of simplicity. The proposed approach actually goes beyond classification problems.

it might be useful to provide the network with examples whose level of complexity progressively increases as long as learning proceeds. However, it is very unlikely to have information on the difficulty of the training examples and, more importantly, if the complexity is determined by humans it might not match the intrinsic difficulty that the machine will face in processing such examples. Alternatively, the value  $\ell$  could be used as an indicator to estimate the difficulty of the data, to exclude the examples with largest loss values or to reduce their contribution in Eq. 3.1, more closely related to SPL (Kumar et al., 2010; Li and Gong, 2017).

Differently from the aforementioned approach, Friendly Training *transforms* the training examples according to the state of the learner, with the aim of discarding the parts of information that are too complex to be handled by the network with the current weights, while preserving what sounds more coherent with the expectations of the current classifier.<sup>3</sup> OFT consists in alternating two distinct optimization phases, that are iterated multiple times. In the first phase, the training data are transformed in order to make them more easily manageable by the current network. The training procedure must determine how data should be simplified according to the way the current network behaves. In the second phase, the network is updated as in CT, but exploiting the simplified data instead of the original ones. The whole procedure is framed in the context of a developmental plan in which the amount of the alteration is progressively reduced as long as time passes, until it completely vanishes. This is inspired by the basic principle of strongly simplifying the data during the early stages of life of the classifier, in order to favour its development, while the extent of transformation is reduced when the classifier improves its skills. Clearly, to deploy a trained classifier that does not rely on altered data, the impact of the simplification must vanish during the training process, exposing the classifier to the original training data after a certain number of steps. Formally, OFT perturbs the training data by estimating the variation  $\delta_i$ ,

$$\tilde{x}_i = x_i + \delta_i, \quad (3.2)$$

for each example  $x_i$ . Given a mini-batch  $\mathcal{B}$ , we indicate with  $\Delta$  the matrix that collects the perturbations associated to the mini-batch examples. In detail, the  $i$ -th row of  $\Delta$  is the perturbation  $\delta_i$  associated with the  $i$ -th example in  $\mathcal{B}$ . For convenience in the notation, we avoid mentioning training epochs in what follows, and we describe the training procedure as the iterative processing of mini-batches of data, updating  $w$  after each of them. Let us denote with  $\gamma \geq 1$  the iteration index. We re-define the aggregated loss  $L$  of Eq. 3.1 by providing the network with  $\tilde{x}$  instead of  $x$ , introduc-

---

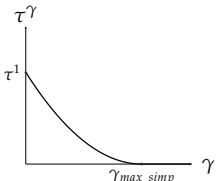
<sup>3</sup>This is significantly different from deciding whether or not to keep a training example, to weigh its contribute in Eq. (3.1), or to re-order the examples. Interestingly, Friendly Training is compatible with (and not necessarily an alternative to) such existing strategies.

ing the dependency on  $\Delta$ , and by adding the iteration index  $\gamma$ ,

$$L(\mathcal{B}^\gamma, \Delta^\gamma, w^\gamma) = \frac{1}{|\mathcal{B}^\gamma|} \sum_{i=1}^{|\mathcal{B}^\gamma|} \ell(f(\tilde{x}_i, w^\gamma), y_i), \quad (3.3)$$

where  $\tilde{x}_i$  is defined as in Eq. 3.2,  $(x_i, y_i) \in \mathcal{B}^\gamma$  (i.e.,  $\mathcal{B}^\gamma$  is the mini-batch at iteration  $\gamma$ ) and  $\delta_i$  is the  $i$ -th row of  $\Delta^\gamma$ . Jointly optimizing Eq. 3.3 with respect to  $\Delta^\gamma$  and  $w^\gamma$  allows the network not only to adapt its weights and biases in order to better cope with the learning criterion, but also to alter the data in  $\mathcal{B}^\gamma$  by translating them in those space regions that can be more easily classified. However, the loss of Eq. 3.3 does not introduce any constraints on each  $\delta_i$ . Hence, the network is free to change the training data without any guarantees that the simplification amount will reduce while learning proceeds. Moreover, differently from  $w^\gamma$ , the set  $\Delta^\gamma$  is specifically associated with the data in  $\mathcal{B}^\gamma$  (i.e., each training example is associated with its own perturbation), meaning that the number of variables of the optimization problem becomes a function of the size of the training data.

We frame Eq. 3.3 in the context of a developmental plan that solves both these issues. First, the system is enforced to reduce the perturbations as long as the number of training iterations increases. If  $\gamma_{max}$  is the maximum number of allowed iterations, we ensure that after  $\gamma_{max\_simp} < \gamma_{max}$  steps the data are not perturbed anymore. Secondly, we remove the dependence of  $\Delta^\gamma$  from  $\gamma$ , introducing an Alternate Optimization scheme in which we decouple the optimization of perturbations and weights. In detail, we consider a single matrix  $\Delta$  for a mini-batch (the total number of rows in  $\Delta$  is equal to the size of a single mini-batch). At the beginning of each training iteration, we keep  $w^\gamma$  fixed, we initialize  $\Delta$  to zeros and then we estimate the appropriate perturbations for the current data in  $\mathcal{B}^\gamma$  by gradient descent over the variable  $\Delta$  (second argument of Eq. 3.3). We indicate with  $\tau^\gamma$  the number of iterative steps of such inner optimization. The value of  $\tau^\gamma$  controls the amount of alteration on the data. For small values of  $\tau^\gamma$  the network will only marginally simplify the data, while for a larger  $\tau^\gamma$  the data alteration will be more aggressive. Nonetheless, it is important to remark that while the degree of the alteration is controlled by  $\tau^\gamma$ , the appearance of the simplification and its spatial distribution can substantially change over time (see Fig. 3.2), reflecting the evolving skills of the learner. The initial value  $\tau^1 \geq 1$  is a fixed hyper-parameter, while we considered a quadratic law to progressively reduce  $\tau$  in function of  $\gamma$ ,

$$\tau^\gamma = \tau^1 \cdot \left[ 1 - \frac{\gamma - 1}{\gamma_{max\_simp}} \right]_+^2. \quad (3.4)$$


being  $[a]_+$  the positive part of  $a$  and  $\gamma \in [1, \gamma_{max}]$ . Afterwards, we update the values of  $w^\gamma$ , given the output  $\Delta$  at the end of the just described inner optimization.

This developmental plan allows the system to adapt the data in order to make the learning procedure less disruptive, especially during the early stages of learning, introducing a smooth optimization path driven by the evolution of  $\tau^\gamma$ .

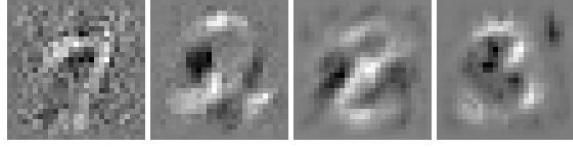


Figure 3.2: OFT, randomly selected perturbations ( $\delta$ ), taken from different stages of the first epoch when using CNN-A (samples from MNIST-BG dataset). Left: perturbations generated close to the beginning of training. Moving toward the right: perturbations generated closer to the end of the epoch. As the system evolves,  $\delta$  mostly involves the portions of image covered by the digits.

The detailed training procedure is formally reported in Alg. 1, and in the fol-

---

**Algorithm 1** Optimization-based Friendly Training.

---

**Input:** Training set  $\mathcal{X}$ , initial weights and biases  $w^1$ , batch size  $b$ , max learning steps  $\gamma_{max}$ , max simplification steps  $\gamma_{max\_simp}$ ,  $\tau^1 \geq 1$ , learning rates  $\alpha > 0$  and  $\eta > 0$ , shared matrix  $\Delta$  with  $b$  rows and  $d$  columns.

**Output:** The final  $w^{\gamma_{max}+1}$ .

- 1: **for**  $\gamma = 1$  to  $\gamma_{max}$  **do**
  - 2:   Sample  $\mathcal{B}^\gamma$  of size  $b$  from  $\mathcal{X}$
  - 3:   Compute  $\tau^\gamma$  following Eq. 3.4.
  - 4:   Set all the entries of  $\Delta$  to 0
  - 5:   **for**  $\tau = 1$  to  $\tau^\gamma$  **do**
  - 6:     Compute  $\Delta_{grad} = \frac{\partial L(\mathcal{B}^\gamma, D, w^\gamma)}{\partial D} \Big|_{D=\Delta}$ , see Eq. 3.3
  - 7:      $\Delta = \Delta - \eta \cdot \Delta_{grad}$
  - 8:   **end for**
  - 9:   Compute  $w_{grad}^\gamma = \frac{\partial L(\mathcal{B}^\gamma, \Delta, v)}{\partial v} \Big|_{v=w^\gamma}$ , see Eq. 3.3
  - 10:    $w^{\gamma+1} = w^\gamma - \alpha \cdot w_{grad}^\gamma$
  - 11: **end for**
  - 12: **return**  $w^{\gamma_{max}+1}$
- 

lowing lines we provide some further details. Notice that while the weight update equation (line 10) can include any existing adaptive learning rate estimation procedure, in our current implementation  $\Delta$  is updated by a fixed small learning rate (line 7). While Alg. 1 formally returns the weights after having completed the last training iteration, as usual, the best configuration of the classifier can be selected by measuring the performance on a validation set, when available. Another important fact to mention is that when the prediction on examples perfectly matches target,

line 6 will return zero gradient, hence no simplifications are performed. In our implementation we slightly relaxed this condition by zeroing the rows of  $\Delta_{grad}$  (line 6) associated to those examples that are classified with a large confidence above a given threshold (see Sec. 3.4), that implements a selective early-stopping condition on the inner optimization.

### 3.3.2 NFT: neural-based model

Despite the novel view introduced by Friendly Training, the instance of OFT is mostly inspired by the basic tools used in the context of Adversarial Training (Zhang et al., 2020a), with a perturbation model that requires a per-example independent optimization procedure. Here we propose to instantiate Friendly Training in a different manner, by considering that there might be some regularities in the way data samples are simplified. This leads to the introduction of a more structured transformation function that is shared by all the examples. This intuition is also motivated by recent studies in Adversarial Machine Learning that exploited perturbation models based on generative networks (Qiu et al., 2020; Xiao et al., 2018), although with the goal of fooling a classifier. Formally, a training sample  $x_i \in \mathbb{R}^d$  is transformed into  $\tilde{x}_i \in \mathbb{R}^d$  by means of the function  $s(x_i, \theta)$ ,

$$\tilde{x}_i = s(x_i, \theta), \quad (3.5)$$

being  $\theta$  a set of learnable parameters, shared by all the examples. We consider the case in which  $s$  is implemented with an additional neural network, also referred to as *auxiliary network*, whose weights and biases are collected in  $\theta$ , and we talk about Neural Friendly Training (NFT). For convenience in the notation, we keep the definition of  $\delta_i$  inherited from Eq. 3.2, i.e.,  $\delta_i = \tilde{x}_i - x_i$ . The term *main network* refers to the network that implements  $f$ , i.e., the classifier, and we report in Fig. 3.3 a sketch of the proposed model.

In order to setup a valid developmental plan, we introduce an augmented criterion by re-defining the risk  $L$  of Eq. 3.1,

$$L(\mathcal{B}, w, \theta) = \frac{1}{|\mathcal{B}|} \sum_{i=1}^{|\mathcal{B}|} \left( \ell(f(\underbrace{s(x_i, \theta)}_{\tilde{x}_i}, w), y_i) + \eta \|\underbrace{s(x_i, \theta) - x_i}_{\delta_i}\|^2 \right), \quad (3.6)$$

where  $(x_i, y_i) \in \mathcal{B}$ , and  $\eta > 0$  is the weight of the squared Euclidean norm of the perturbation  $\delta_i$ . We indicate with  $\gamma \geq 1$  the NFT iteration index, where each iteration consists of the two aforementioned phases. In the first phase, the auxiliary network is updated by minimizing Eq. 3.6 with respect to  $\theta$ , keeping the main network fixed.



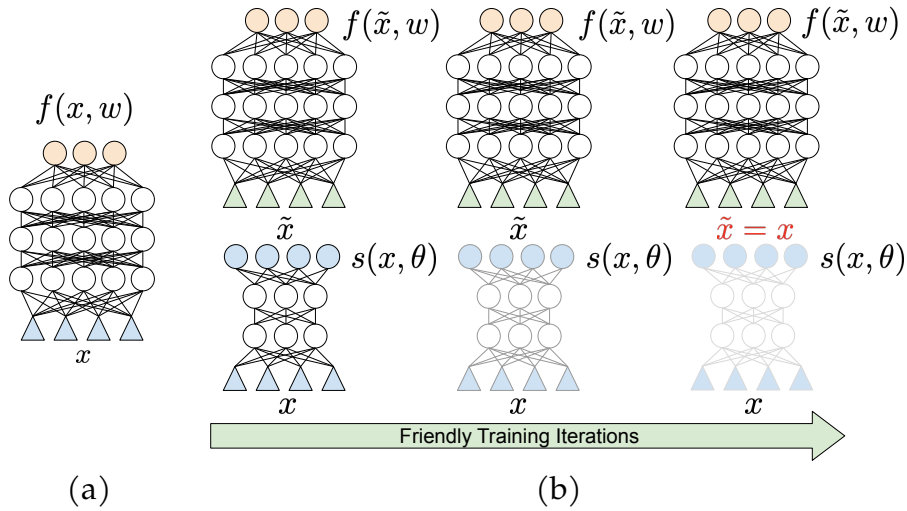
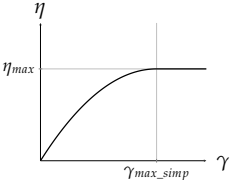


Figure 3.3: (a) Classic deep network. (b) Neural Friendly Training (NFT): main deep network (top) and auxiliary network (bottom). The auxiliary network learns how to simplify the data  $x$ , while the main network learns the classification task exploiting the simplified data  $\tilde{x}$ . As long as training proceeds, the effect of the auxiliary network is progressively reduced, until it vanishes (and it is removed).

In the second phase, the auxiliary network has the sole role of transforming the data, while the main network is updated by minimizing Eq. 3.6 with respect to  $w$ . If all the training data is used in this phase, then  $\gamma$  boils down to the epoch index (that is the case we considered in the experiments). If  $\gamma_{max}$  is the maximum number of NFT iterations, we ensure that after  $\gamma_{max\_simp} < \gamma_{max}$  steps the data are not perturbed anymore. In order to progressively reduce the perturbation level, we increase the value of  $\eta$  in Eq. 3.6. It is significant to point out that  $\eta$  in NFT plays a similar role as  $\tau^\gamma$  in OFT (Eq. 3.4), but it has a complementary evolution over time. For a large  $\eta$ , NFT will strongly penalize the norm of  $\delta_i$ , becoming the dominant term in the optimization process of the auxiliary network, enforcing the net to keep  $\delta_i$  small. We indicate with  $\eta_{max}$  the maximum possible value of  $\eta$ , and at each step  $\gamma$  of the developmental process we compute  $\eta$  using the following law, being  $[a]_+$  the positive part of  $a$ ,

$$\eta = \eta_{max} \left( 1 - \left[ 1 - \frac{\gamma - 1}{\gamma_{max\_simp} - 1} \right]_+^2 \right) \quad (3.7)$$


The graph shows the relationship between  $\eta$  and  $\gamma$ . The vertical axis is labeled  $\eta$  and has a tick mark at  $\eta_{max}$ . The horizontal axis is labeled  $\gamma$  and has a tick mark at  $\gamma_{max\_simp}$ . The curve starts at the origin (0,0) and increases monotonically, following a sigmoidal shape, eventually plateauing at  $\eta_{max}$  for  $\gamma \geq \gamma_{max\_simp}$ .

where  $\gamma \in [1, \gamma_{max}]$ . At  $\gamma_{max\_simp}$  iterations, the penalty on  $\|\delta_i\|^2$  will reach its maximum weighting. While this encourages the function  $s(\cdot, \theta)$  to get closer to the identity function, we have no formal guarantee that optimization will effectively push the perturbation to zero. For this reason, after  $\gamma_{max\_simp}$  iterations we drop the auxiliary

network, exposing the system to the original training data. The developmental plan on  $\eta$  favours a smooth transition between the setting in which the auxiliary network is used and when it is removed.

The training procedure is detailed in Alg. 2, and in the following lines we provide

---

**Algorithm 2** Neural Friendly Training.

---

**Input:** Training set  $\mathcal{X}$ , initial weights and biases  $w$ , batch size  $b$ , max FT steps  $\gamma_{max}$ , max simplification steps  $\gamma_{max\_simp}$ ,  $\eta_{max} > 0$ , learning rates  $\alpha > 0$  and  $\beta > 0$ .

**Output:** The final  $w$ .

```

1: for  $\gamma = 1$  to  $\gamma_{max}$  do
2:   Compute  $\eta$  following Eq. 3.7
3:   if  $\gamma > 1$  and  $\gamma \leq \gamma_{max\_simp}$  then
4:      $s \leftarrow \text{auxiliary\_net}(\cdot, \theta)$ 
5:     Sample a set of minibatches  $B = \{\mathcal{B}_z\}$  from  $\mathcal{X}$ 
6:     for each mini-batch  $\mathcal{B}_z \in B$  do
7:       Compute  $\nabla_{\theta} = \left. \frac{\partial L(\mathcal{B}_z, w, h)}{\partial h} \right|_{h=\theta}$ , see Eq. 3.6
8:        $\theta = \theta - \beta \cdot \nabla_{\theta}$ 
9:     end for
10:  else
11:     $s \leftarrow I(\cdot)$ 
12:  end if
13:  Sample a set of minibatches  $B = \{\mathcal{B}_z\}$  from  $\mathcal{X}$ 
14:  for each mini-batch  $\mathcal{B}_z \in B$  do
15:    Compute  $\nabla_w = \left. \frac{\partial L(\mathcal{B}_z, h, \theta)}{\partial h} \right|_{h=w}$ , see Eq. 3.6
16:     $w = w - \alpha \cdot \nabla_w$ 
17:  end for
18: end for
19: return  $w$ 

```

FIRST PHASE: update  $s(\cdot, \theta)$

SECOND PHASE: update  $f(\cdot, w)$

---

some further details. The auxiliary network is not updated during the first iteration ( $\gamma = 1$ ), since the main network is still in its initial/random state. After  $\gamma_{max\_simp}$  iterations, the auxiliary network is replaced by the identity function  $I(\cdot)$  (line 10). Notice that the weight update equations (line 8 and line 16) can include any existing adaptive learning rate estimation procedures. In our current implementation we are using the Adam optimizer with learning rates  $\alpha$  and  $\beta$  (Kingma and Ba, 2015), unless differently stated. While Alg. 2 formally returns the weights after having completed the last training iteration, as usual, the best configuration of the classifier can be selected by measuring the performance on a validation set (bypassing the auxiliary net at inference time).

We qualitatively show the behavior of the proposed training strategy in the toy example of Fig. 6.6. A very simple network with one hidden layer (5 neurons with hyperbolic tangent activation function) is trained on the popular *two-moon* dataset

(two classes, 300 examples), optimized by Adam with mini-batch of size 64. The auxiliary network alters the training data in order to make them almost linearly separable during the early iterations. Then, the data distribution progressively moves toward the original configuration, and the decision boundary of the main classifier smoothly follows the data. In the last plot, the auxiliary network has been dropped and examples are located at their original positions in final stages of developmental plan.

Of course, NFT increases the complexity of each training step, due to the extra projection computed by the auxiliary network in the forward stage of the classifier and due to the first phase of Alg. 2. The actual additional computational burden of NFT with respect to CT depends on the architecture of the auxiliary network and on the number of sampled mini-batches in the first phase. Moreover, instead of Eq. 3.7, different developmental plans could be selected to more quickly reduce the simplification and eventually drop the auxiliary network before the end of training. When comparing NFT and OFT we can see that, from the storage point of view, NFT needs to memorize a new network and the associated intermediate variables for optimization purposes, while OFT only requires a new set of variables to store the delta terms. However, from the computational point of view, for each example  $x_i$ , OFT performs  $\tau \geq 1$  iterations to update the perturbation  $\delta_i$ , that implies  $\tau$  inference steps on the main network (see Alg. 1). Differently, NFT does not require any inner example-wise iterative procedures, since it exploits the outcome of first phase in Alg. 2. The inference time in the auxiliary network determines the concrete variations in terms of computational times with respect to OFT. In our experience, on average, training with NFT took similar times to the ones of OFT, since  $\tau$  (in OFT) gets reduced as time passes and we early stopped the inner OFT iterations.

### 3.4 Experiments

We carried out a detailed experimental activity aimed at evaluating how NFT behaves when compared to OFT. We considered the same experimental setting for the two methods, mostly focusing on visual data (digit and shape recognition, general image classification), with some results on textual data (sentiment analysis) to prove that in principle our method can be used for non-vision tasks. We also performed an in-depth analysis on NFT (Sec. 3.5).

We considered four neural classifiers,<sup>4</sup> that consist in two feed-forward Fully-Connected multi-layer perceptrons, referred to as FC-A and FC-B, two Convolutional Neural Networks, named CNN-A and CNN-B, and we also tested a ResNet18 (He et al., 2016) in one of the following experiences, motivated by related work (Wu

<sup>4</sup>Code available at <https://sailab.diism.unisi.it/friendly>.

et al., 2021). FC-A is a simple one-hidden-layer network with hyperbolic tangent activations (10 hidden neurons), while FC-B is deeper and larger model, with 5 hidden layers (2500-2000-1500-1000-500 neurons), batch normalization and ReLU activations. CNN-A consists of 2 convolutional layers, max pooling, dropout and 2 fully connected layers, while CNN-B is deeper (4 convolutional layers). Both of them exploit ReLU activation functions on the convolutional feature maps (32-64 filters in CNN-A, 32-48-64-64 filters in CNN-B) and on the fully connected layers (9216-128 neurons for CNN-A, 5184-128 neurons for CNN-B). Unless differently stated, learning of weights and biases is driven by the minimization of the cross-entropy loss, exploiting the Adam optimizer (Kingma and Ba, 2015) with mini-batches of size 32.

The auxiliary network was selected depending on the type of data that it is expected to simplify. The output layer has the same size of the input one and linear activation. In the case of image data, the auxiliary network is inspired by U-Net (Ronneberger et al., 2015). U-Net progressively down-samples the image, encoding the context information into the convolutional feature maps, and then it up-samples and transforms the data until it matches the input size, also exploiting skip connections.<sup>5</sup> In the case of 1-dim data, we used a fully-connected auxiliary net with 256 hidden neurons.

In all the experiments, networks were randomly initialized, providing the exact same initialization to both OFT/NFT and CT, and we report results averaged over 3 runs, corresponding to 3 different instances of the initialization process. For each OFT/NFT iteration, we sampled non-overlapping mini-batches until all the training data were considered, so that  $\gamma$  is also the epoch index. We selected a large number of epochs  $\gamma_{max}$  which we found to be sufficient to obtain a stable configuration of the weights in preliminary experiences (detailed below), and the reported metrics are about the model with the lowest validation error obtained during training. The error rate was selected as the main metric, since it is one of the most common and simple measure in classification problems. We performed some preliminary experiments to determine the optimal Adam learning rate in the case of CT. Then, we tuned the OFT/NFT hyper-parameters by grid search (detailed below).

In order to have a more extensive experimental validation of the proposed approaches, we include an additional baseline: Easy-Examples First (EEF). When using this training method, mini-batch examples are sorted by loss value in ascending order and only the first  $k$  of them are used to calculate the gradients. At the beginning,  $k = 1$ , so that only 1 example per mini-batch is kept, then  $k$  grows with the training iterations following the same dynamics of OFT. The criterion implemented

---

<sup>5</sup>Code: <https://github.com/milesial/Pytorch-UNet>. In the down-sampling part, 2 initial conv. layers encode the image into  $n_f$  feature maps. Then,  $v$  down-sampling blocks (each of them composed of maxpooling and 2 conv. layers) are followed by  $v$  up-sampling blocks (each of them composed of bilinear upscaling and 2 conv. layers). We considered  $v \in \{1, 2\}$ , and  $n_f \in \{64, 96, 128\}$ .

by EEF can be identified as a basic instance of CL, closely related to OFT (i.e., it has the same temporal dynamics, but examples are not altered and only sub-selected).

### 3.4.1 Advanced digit and shape recognition

The collection of datasets presented in (Larochelle et al., 2007) is about 10-class digit recognition problems and shape-based binary classification tasks ( $28 \times 28$ , grayscale). In detail, `MNIST-ROT` consists of MNIST digits rotated by a random angle, while `MNIST-BG` features non-uniform backgrounds extracted by some random images, and `MNIST-ROT-BG` combines the factors of variations of the first two datasets. In `RECTANGLES-BG` we find representations of rectangles, that might be wide or tall, with the inner and outer regions eventually filled with patches taken from other images, while `CONVEX` is about convex or non-convex white regions on a black background. Datasets ( $\approx 60k$  samples) are already divided into training, validation and test set. We compared the test error rates of the FC-A/B and CNN-A/B models in CT, OFT/NFT, and also using the CL-inspired data sorting policy EEF, with the same temporal dynamics of OFT. Experiments are executed for  $\gamma_{max} = 200$  epochs, and we selected the model with the lowest validation error considering  $\eta_{max} \in \{500, 1000, 2000\}$ ,  $\gamma_{max\_simp} \in \{0.25, 0.5, 0.85\} \cdot \gamma_{max}$ ,  $\beta \in \{10^{-5}, 10^{-4}, 5 \cdot 10^{-4}\}$ .

Tab. 3.1 reports the test error rate of the different models, where other baseline results (overcome by OFT/NFT), exploiting different types of classifier, can be found in (Marullo et al., 2021). Our analysis starts by confirming that the family of Friendly Training algorithms (being them neural or not) very frequently shows better results than CT and EEF. Moreover, the proposed NFT almost always improves the results of OFT, supporting the idea of using an auxiliary network to capture regularities in the simplification process. In the case of CNN-A and CNN-B, the error rate of NFT is lower than in OFT, with the exception of `RECTANGLES-BG`, even though NFT reported a pretty large standard deviation. In fully-connected architectures FC-A and FC-B, we still observe a positive impact of NFT, that usually beats OFT. However, the improvement over CT can be appreciated in a less evident or more sparse manner. As a matter of fact, these architectures are less appropriate than CNNs to handle image data. However, it is still interesting to see how FC-B benefits from the auxiliary network introduced in NFT, where the latter is indeed a convolutional model. Overall, results show that using an auxiliary network is better than independently estimating the perturbation offsets of each example, confirming the capability of the network to learn shared facets of the simplification process.

### 3.4.2 Sentiment analysis

We investigate how OFT/NFT behave in Natural Language Processing considering the task of Sentiment Analysis (positive/negative polarity). We selected two

		MNIST-BG	MNIST-ROT-BG	MNIST-ROT	RECTANGLES-BG	CONVEX
FC-A	CT	28.34±0.09	64.06±0.31	43.16±0.51	24.31±0.21	33.91±0.44
	EEF	<b>28.18</b> ±0.47	64.27±0.19	43.91±0.73	24.48±0.11	<b>33.17</b> ±0.93
	OFT	28.66±0.06	64.14±0.36	43.24±0.43	24.64±0.37	34.38±0.22
	NFT	<b>28.15</b> ±0.04	64.55±0.14	<b>42.96</b> ±0.58	24.57±0.19	34.25±1.03
FC-B	CT	21.06±0.39	51.71±0.79	10.13±0.27	25.10±0.20	27.24±0.05
	EEF	21.38±0.18	52.95±0.63	<b>10.04</b> ±0.17	<b>24.84</b> ±0.32	28.21±0.96
	OFT	21.74±0.26	<b>51.02</b> ±0.07	11.19±0.37	<b>24.14</b> ±0.53	27.49±0.07
	NFT	<b>20.91</b> ±0.52	<b>50.20</b> ±0.16	<b>10.09</b> ±0.32	<b>25.09</b> ±0.09	<b>26.81</b> ±0.15
CNN-A	CT	7.25±0.16	29.05±0.45	7.48±0.14	9.86±0.32	8.24±0.09
	EEF	<b>7.02</b> ±0.08	29.12±0.34	7.61±0.22	12.82±0.70	8.72±0.74
	OFT	<b>6.80</b> ±0.19	<b>28.74</b> ±0.29	<b>7.36</b> ±0.06	<b>9.72</b> ±0.20	8.59±1.44
	NFT	<b>6.59</b> ±0.09	<b>28.67</b> ±0.35	<b>7.17</b> ±0.17	10.99±1.89	<b>8.03</b> ±0.23
CNN-B	CT	5.15±0.15	23.05±0.21	6.58±0.06	8.10±1.90	3.01±0.41
	EEF	<b>4.82</b> ±0.19	<b>22.89</b> ±0.49	7.02±0.28	8.35±1.01	3.75±0.58
	OFT	<b>5.03</b> ±0.11	<b>22.81</b> ±0.36	6.95±0.12	<b>7.32</b> ±1.31	<b>2.87</b> ±0.42
	NFT	<b>4.96</b> ±0.34	<b>22.22</b> ±0.62	<b>6.48</b> ±0.25	<b>6.27</b> ±0.62	<b>2.78</b> ±0.34

Table 3.1: Comparison of different classifiers (FC-A, FC-B, CNN-A, CNN-B) and learning algorithms (CT, EEF, OFT and NFT). Test error and standard deviation over 3 runs are reported. For each architecture, those results that improve the CT case are in bold.

datasets and considered different representations of the examples. The first dataset is `IMDB` (Maas et al., 2011), also known as Large Movie Review Dataset, that is a collection of 50k highly-polar reviews from the IMDB database. We considered a vocabulary of the most frequent 20k words and TF-IDF (Jones, 1972) representation of each review. The second dataset, `WINES` (Thoutt, 2017), collects 130k wine reviews scored in  $[80, 100]$ , that we divided into two classes, i.e.,  $[80, 90)$  vs.  $[90, 100]$ . In this case, in order to get more general findings, we chose a different text representation, exploiting a pretrained Transformer-based architecture. We preprocessed the input samples with a pretrained DistilRoBERTa (Reimers and Gurevych, 2019) model, applying average pooling to compute dense representations of size 768 for each review.

We trained the deeper fully-connected architecture, FC-B, for 30 epochs. NFT hyper-parameters<sup>6</sup> were selected in  $\eta_{max} \in \{10, 100, 500, 1000, 2000\}$ ,  $\gamma_{max\_simp} \in \{0.05, 0.1, 0.25, 0.85, 0.5\} \cdot \gamma_{max}$ ,  $\beta \in \{10^{-5}, 10^{-4}, 5 \cdot 10^{-4}\}$ .

As reported in Tab. 3.2 (top), the performance of CT is consistently improved by NFT, achieving lower error rates in both the datasets (and representations). The sentence classification task appears to be slightly more difficult in `WINES`. This is

<sup>6</sup>Concerning OFT, the hyper-parameter grids are available in supplementary material, published at <https://sailab.diism.unisi.it/friendly>.

probably due to the fact that wine reviews are less polarized, being them all highly scored. Concerning IMDB, the superiority of NFT over CT and also OFT is evident. Overall, these results confirm the versatility of NFT.

IMDB		WINES	
FC-B CT	13.27 $\pm$ 0.19	FC-B CT	17.38 $\pm$ 0.15
FC-B OFT	13.66 $\pm$ 0.69	FC-B OFT	<b>17.07</b> $\pm$ 0.11
FC-B NFT	<b>11.93</b> $\pm$ 0.09	FC-B NFT	<b>17.15</b> $\pm$ 0.12
CIFAR-10		CIFAR-10-N10	
CNN-B CT	29.75 $\pm$ 0.37	ResNet CT	9.30 $\pm$ 0.16
CNN-B OFT	30.19 $\pm$ 0.53	ResNet OFT	<b>8.92</b> $\pm$ 0.23
CNN-B NFT	<b>29.00</b> $\pm$ 0.36	ResNet NFT	<b>8.10</b> $\pm$ 0.19

Table 3.2: Comparison of classifiers with different architectures and learning algorithms (CT, OFT, NFT) – data for Sentiment Analysis (top) and Image Classification (bottom). Mean test error is reported with standard deviation. Results improving CT are in bold.

### 3.4.3 Image classification

CIFAR-10 (Krizhevsky, 2009) is a popular Image Classification dataset, consisting of 60k  $32 \times 32$  color images from 10 different classes. We divided the original training data into training and validation sets (10k examples used as validation set), and we initially evaluated NFT using the previously described generic CNN-B architecture. Tab. 3.2 (bottom-left) shows that while we are not able to improve the results of CT using OFT, NFT slightly improves the quality of the network, reducing the error rate and further confirming its benefits.

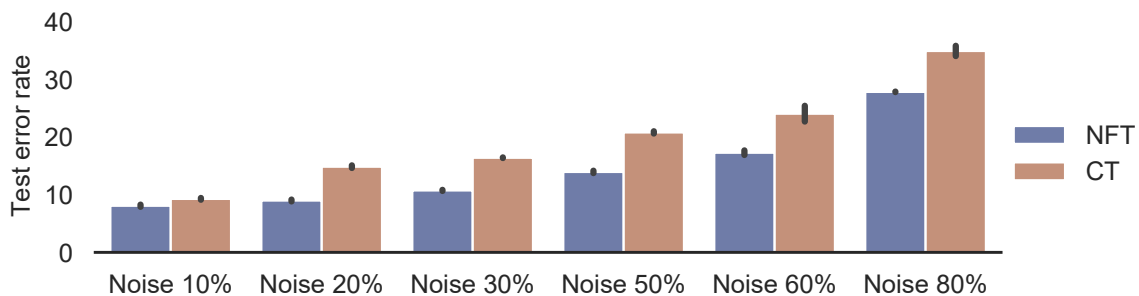


Figure 3.4: ResNet18 on CIFAR-10 dataset for different amounts of noisy labels. Error bars include standard dev.

However, state-of-the-art convolutional networks specifically designed/tuned for CIFAR-10 usually achieve lower error rates, so that we decided to perform a more

specific experimental activity. In particular, we considered ResNet18 (He et al., 2016), inheriting all the carefully selected optimization parameters and tricks that yield state-of-the-art results in CIFAR-10.<sup>7</sup> Since OFT/NFT bring marginal benefits over CT, we designed a more challenging condition following the setup of recently published CL activity (Wu et al., 2021). We introduced some noise by randomly permuting 10% of the target labels, generating what we will refer to as CIFAR-10-N10. We trained the network for 250 epochs, and reported results in Tab. 3.2 (bottom-right). NFT hyper-parameters were selected in  $\eta_{max} \in \{500, 1000, 2000\}$ ,  $\gamma_{max\_simp} \in \{0.25, 0.5, 0.7\} \cdot \gamma_{max}$ ,  $\beta \in \{10^{-4}, 5 \cdot 10^{-4}\}$ . The learning rate scheduler is applied starting from  $\gamma_{max\_simp}$  with an initial learning rate which is  $0.1 \cdot \alpha$ . We observe that NFT effectively helps also when dealing with this type of network. While OFT also carries a small improvement, it is far from the one obtained by NFT. We further investigated this result by varying the amount of noise injected into the training labels. Fig. 3.4 compares CT and NFT for different noise levels, up to 80%. Interestingly, the impact of NFT becomes more and more evident, gaining  $\approx 8\%$  in strongly noisy environments, suggesting that data simplification helps the main network to better discard misleading and noisy information.

### 3.5 Discussion

We qualitatively compared NFT and OFT in the MNIST-BG dataset of Section 3.4.1, in which the important information is known (the digits), since the background is uncorrelated with the target. We mostly considered the CNN-A model, for which NFT led to the most significant improvements with respect to CT (Tab. 3.1). In Fig. 3.5 we show how examples are affected when using an auxiliary network (bottom - NFT) or when independent transformations are estimated for each example through a gradient-based procedure (top - OFT). Modelling the transformation function with a neural model leads to qualitatively different behavior. We observe that OFT yields structured perturbations only when paired with CNN-A, emphasizing the digit areas. Differently NFT shows more natural perturbation patterns, removing distracting cues (background). Basically, the convolutional auxiliary net leads to transformations with much more detailed awareness of the visual structures.

In Fig. 3.6, we report the evolution of test error rate during the training epochs (MNIST-BG, CNN-A), comparing NFT and CT. The developmental plan reduces the impact of the perturbation until epoch 175 (afterwards, data are not altered anymore). The small bump right before such epoch is due to the final transition from altered to original data. The test error of NFT is higher than the one of CT when

<sup>7</sup>Stochastic Gradient Descent (learning rate 0.1 with cosine annealing learning rate scheduler) with momentum (0.9) and weight decay ( $5 \cdot 10^{-4}$ ), mini-batches of size 128, data augmentation – see <https://github.com/kuangliu/pytorch-cifar>.



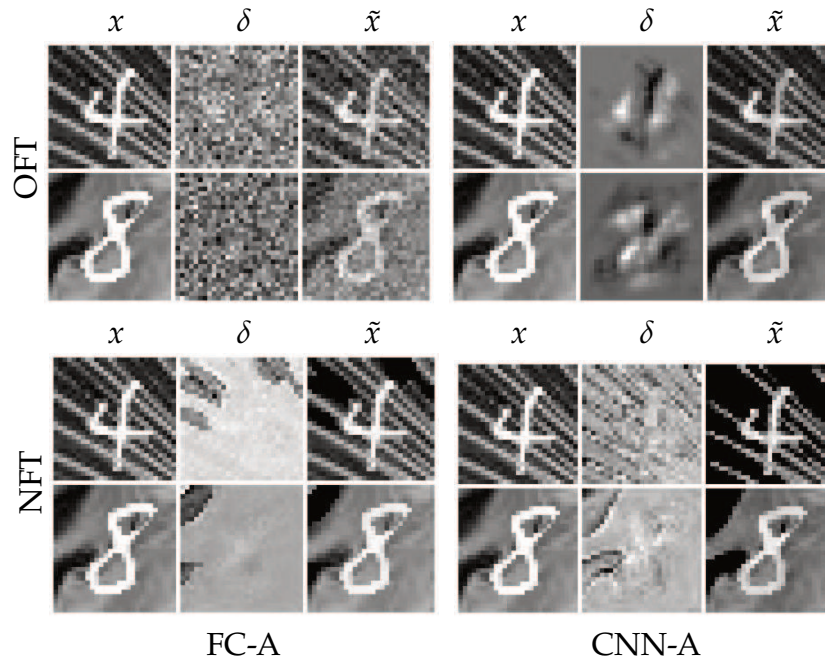


Figure 3.5: MNIST-BG. Original data  $x$ , perturbation  $\delta$  (normalized) and resulting “simplified” images  $\tilde{x}$  for FC-A and CNN-A at the end of the 1st epoch. At first glance, some simplified images are hardly distinguishable from the original samples. Top: OFT. Bottom: NFT.

data are altered, as expected, while it becomes lower when the auxiliary network is dropped. On the other hand, fitting training data is easier during the early epochs in NFT, due to the simplification process.

We also evaluated the sensitivity of the system to some hyper-parameters of NFT, keeping the main network fixed. In Fig. 3.7, we report the test error of CNN-A, MNIST-BACK-IMAGE dataset, for different configurations of  $\eta_{max}$ ,  $\frac{\gamma_{max\_simp}}{\gamma_{max}}$ ,  $n_f$ ,  $\beta$ . In particular, after having selected a sample run that is pretty representative of the general trend we observed in the experiments, we changed one of the aforementioned parameters and computed the error rate. Large values of  $\eta_{max}$  reduce the freedom of auxiliary network in learning the transformation function.

Similarly, a short developmental plan with a small  $\frac{\gamma_{max\_simp}}{\gamma_{max}}$  does not allow the main network to benefit from the progressively simplified data. In general, we did not experience a very significant sensitivity to the variations of  $n_f$ , and 64 features turned out to be fine in most of the experiments, with some cases in which moving to 96 was slightly preferable, as in the one we are showing in Fig. 3.7. Although in a fine-grained grid of values, we found that larger  $\beta$  helped the auxiliary network to more quickly develop meaningful transformations. As a side note, we report that NFT was  $\approx 1.5\times$  slower than CT, on average—see comments in related part of Sec. 3.3,

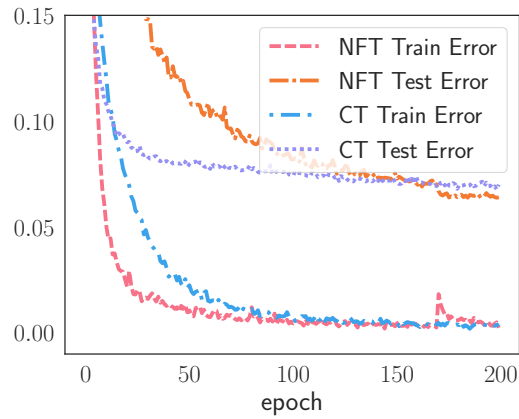


Figure 3.6: Training and test error rates for NFT and CT on a single run – MNIST-BACK-IMAGE, CNN-A (best viewed in colors). The auxiliary network is dropped at epoch 175. The training error of NFT is initially lower than in the case of CT since the auxiliary network simplifies the data. Differently, the test error is initially larger, since the test set is not simplified. As training proceeds, the simplification vanishes and the test data become aligned with the training ones.

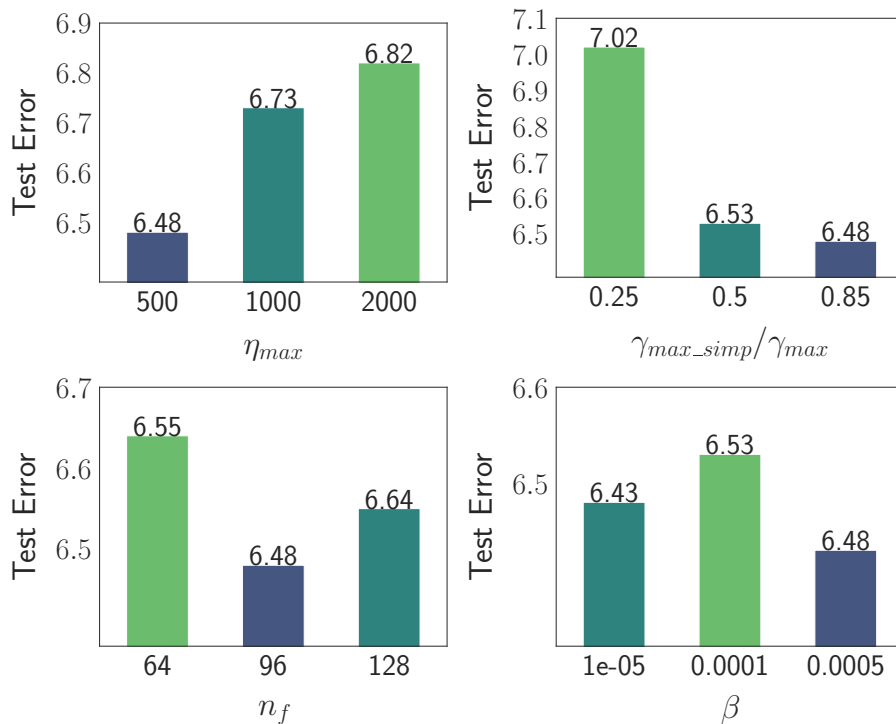


Figure 3.7: Test error under different configuration of the NFT hyper-parameters, CNN-A architecture.

### 3.5.1 Remarks

To summarize, in this chapter we presented a novel approach to Curriculum Learning, according to which training data are altered in order to improve the learning procedure of a neural network-based classifier. Thanks to a progressive developmental plan, the classifier implicitly learns from examples that better match its current expectations, reducing the impact of difficult examples or noisy data during early training. We showed that NFT (the neural-based implementation, see Sec. 3.3.2) systematically outperforms OFT (the initial direct-optimization implementation, see Sec. 3.3.1). The auxiliary neural network is dropped at the end of the training process, so that the computational cost at inference time is unvaried. An extensive experimental evaluation showed the benefits of Neural Friendly Training, with improved generalization skills with respect to standard training.

# Chapter 4

## Input tuning for Continual Learning

The intrinsic difficulty in adapting deep learning models to non-stationary environments limits the applicability of neural networks to real-world tasks. This issue is critical in practical supervised continual learning settings, such as the ones in which a pre-trained model computes projections toward a latent space where different task predictors are sequentially learned over time. As a matter of fact, incrementally fine-tuning the whole model to better adapt to new tasks usually results in catastrophic forgetting, with decreasing performance over the past experiences and loss of valuable knowledge from the pre-training stage. In this chapter, we propose a novel strategy to make the fine-tuning procedure more effective, by avoiding to update the pre-trained part of the network and learning—apart from the usual classification head—also a set of newly-introduced learnable parameters that are responsible for transforming the input data. We term this method as *Continual Input Tuning* (CIT), to propose an alternative to conventional fine-tuning but also to underscore that we preprocess the samples in the input space through a trainable transformation that is *tuned* over time with reduced forgetting. This concept is inspired by the dynamic input manipulations of Ch. 3, but here the learner is already equipped with some knowledge and faces an uncontrolled sequence of tasks. This process allows the network to effectively leverage the pre-training knowledge and find a good trade-off between plasticity and stability with modest computational efforts, thus especially suitable for on-the-edge settings. Our experiments on four image classification problems in a continual learning setting confirm the quality of the proposed approach when compared to several fine-tuning procedures and to popular Continual Learning methods.

### 4.1 Introduction

The outstanding performance achieved by Machine Learning solutions in a vast variety of fields (Brown et al., 2020) and well-defined tasks (Russakovsky et al., 2015)

is usually restricted to a very specific setting, where it's assumed that all training data is available from the start and sampled from a static distribution in an independent manner (i.i.d.). This scenario does not contemplate the case in which neural models are progressively adapted to novel data that are sampled over time from non-stationary distributions. In such cases, a huge performance drop on old data is typically noticed when adapting weights (French, 1999). Recently, these limitations have gained wider attention and novel models that are designed to learn over time have started to emerge (see Sec. 2.2 for an introduction to Continual Learning—henceforth, CL).

At the same time, intelligent edge devices, including sensors, actuators, robotic platforms, etc., with modest computational resources, are becoming ubiquitous and there is a growing need of Machine Learning-driven processing capabilities for perception, understanding and personalization in the Internet of Things. Indeed, such devices are capable of acquiring continuous data streams (Murshed et al., 2021), that could be processed with CL-based solutions running on the edge devices themselves. Several challenges must be faced when deploying CL methods to edge devices. First of all, typical state-of-the-art neural architectures with many encoding layers (Vaswani et al., 2017) might be of limited applicability due to scarcity of memory and computational capabilities. While offloading to the cloud might be a simple workaround, it raises privacy concerns and complexity issues. Moreover, most of the state-of-the-art CL methods involve rehearsal procedures (De Lange et al., 2021), i.e., re-visiting some exemplars of past concepts in order to “refresh” the knowledge of the model. This opens to new issues concerning storage capacity and, again, privacy (Pellegrini et al., 2020), given that it introduces the need of long-term storage of training data. Methods based on latent replay (rather than input replay) achieve better compression and obfuscate potentially private data, but storage issues are not solved and privacy preservation may be totally unreliable (Mahendran and Vedaldi, 2015). Another issue with the current CL research concerns the limited efforts spent in evaluating the performance of CL methods in realistic applications.

In the context of edge devices with limited computational capabilities, it seems reasonable to start from networks pre-trained on large data collections, since they are powerful tools to compute informed latent representations and they allow to reduce training efforts. However, their embedded knowledge might be significantly lost when progressively fine-tuning the network to novel domains in a CL fashion, unless rehearsal is performed (Ostapenko et al., 2022). An alternative to fine-tuning has recently become popular in the Machine Learning community, i.e., learning parameters that affect the model input with the goal of conditioning the network rather than changing its internal weights. These Prompt Tuning models (Li and Liang, 2021; Lester et al., 2021; Jia et al., 2022) were conceived to foster the exploitation of very-large-scale networks, in order to effectively leverage their capabilities

in specific downstream tasks. However, most of the work in this line focuses on large Transformer models and the connection with CL is not deeply investigated (Wang et al., 2022b). Motivated by these considerations, we propose and investigate the appropriateness of different non-replay tuning options when facing a CL problem based on a pre-trained network, frequently referred to as *backbone*, focusing on methods that require lower computational efforts and no-rehearsal, well suited for edge devices. In particular, we propose *Continual Input Tuning* (CIT), based on an alternative form of Prompt Tuning, in order to efficiently adapt the model to new data and achieve a good trade-off between plasticity and stability. According to the taxonomies introduced in Sec. 2.2 for Continual Learning methods, CIT can be considered a representation-based method, since it relies on the exploitation of frozen pretrained models, but also an architecture-based method since it adds special parameters to the computational model. We provide an experimental analysis conducted on different datasets available in the related literature, comparing several fine-tuning procedures and well-known continual learning methods. The contributions of this chapter are the following ones: (1) we propose the adoption of Continual Input Tuning procedures to better leverage pre-trained backbones in CL; (2) we experimentally evaluate the impact of fine-tuning on common CL benchmarks, providing reference results; (3) we investigate and show the benefits of Continual Input Tuning in a classic continual setting, with edge-friendly neural architectures, both for small and large domain shifts.

## 4.2 Relation with existing works

The wide availability of pre-trained models offers several opportunities to transfer their knowledge to specific downstream tasks. The simplest approach consists in fine-tuning the models on the novel task data. This is typically demanding in terms of resources, especially in the case of large-scale models, due to the memory occupation (gradients as well as activations) and the operations in the weight update routine. The generic notion of pre-training has been shown to implicitly make some CL problems easier (Mehta et al., 2023), following the intuition that it is more likely to end up in good suboptimal minima compared to randomly-initialized models, due to the skills learned in the pre-training stage. Of course, an inappropriate fine-tuning procedure might yield a model strongly focused on the novel task, losing the advantage from the previously learned knowledge. However, while the limits of transferring pre-trained models to downstream tasks have been recently studied (Abnar et al., 2022), when it comes to specifically studying or evaluating the concrete impact of adapting pre-trained models to a CL context, the scientific literature is relatively scarce (Ostapenko et al., 2022). Ramasesh et al. (2022) have pointed out that resistance to forgetting consistently scales with the network size when em-

ploying pre-trained models. Still, several questions remain unanswered, especially concerning smaller-scale models in computationally-restricted environments, as the ones we study in this chapter. Another line of work (Hu et al., 2022; Cossu et al., 2022) replaces the pre-training step with a continual procedure, especially in the self-supervised setting. While this approach paves the way for the exploitation of decentralized and streaming data in a variety of contexts, it is not focused on the practical scenario in which downstream tasks are learnt in an incremental fashion.

Prompt Tuning methodologies (Li and Liang, 2021; Lester et al., 2021; An et al., 2022), that were originally conceived in Natural Language Processing (NLP), open a new perspective on how to adapt a pre-trained model to a novel environment, with respect to the one of the pre-training stage. Prompt Tuning consists in learning parameters that are actually part of the input stage, in order to discover the proper way to condition the model, rather than changing the values of its internal weights or learning additional internal parameters, as in the case of Adapters (Pfeiffer et al., 2020). Such technique has been mostly applied to Transformer models (Vaswani et al., 2017), both for language tasks and vision tasks. In this chapter we focus on a specific type of tuning that is not restricted to Transformer models, and that we study in the context of CL.

Tweaking the input space (*input tuning*) to alter the behavior of neural networks has already been investigated for a variety of purposes. For instance, with this strategy, Elsayed et al. (2019) successfully managed to re-program a trained model to perform on a very different task. It is now important to draw a connection with what we discussed in Ch. 3, where we presented a curriculum-learning inspired methodology to effectively learn from a continuous set of learning problems with gradually increasing complexity. Recently, the effectiveness of learning simple transformations in the input space has been studied in the context of transfer learning for image classification (Jia et al., 2022), and it has been shown that learning parameters to transform the input data is a competitive approach to deal with large pre-trained models, especially Transformer-based ones. Researchers are starting to apply this intuition to the CL perspective (Wang et al., 2022b), but they are mostly focused on rather large ViT (Dosovitskiy et al., 2021) models. Moreover, they heavily rely on the capability of the considered model to extract global semantically-rich representations to guide the input transformation, while they lack investigation of the non-Transformer case. We are going to study how altering the input data by means of newly introduced learnable parameters can yield efficient and computationally affordable CL, starting from a pre-trained backbone.

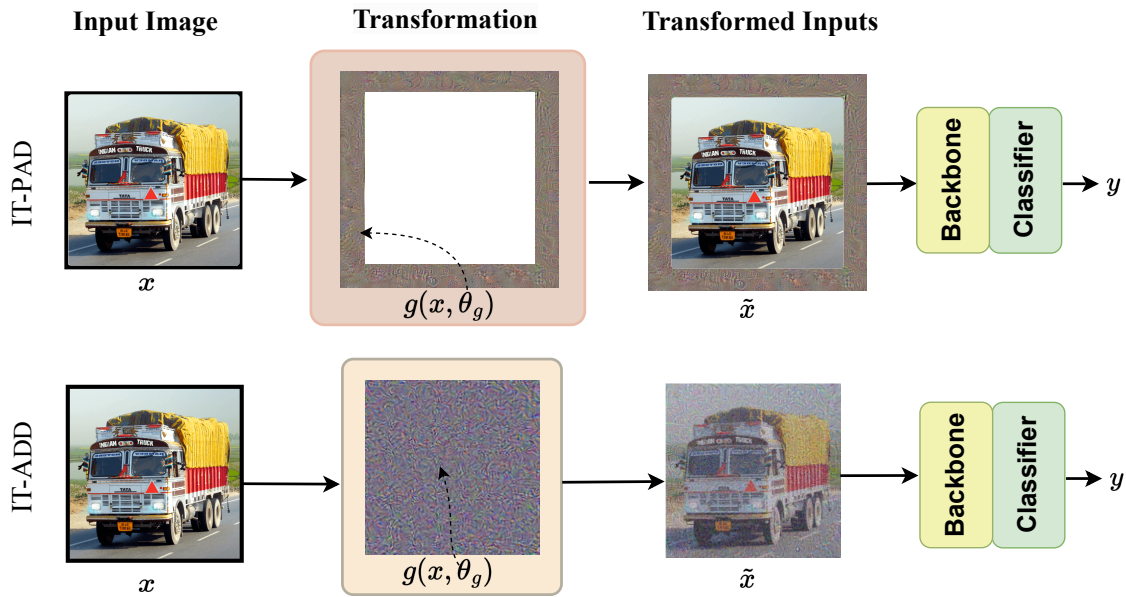


Figure 4.1: Sketch of the proposed input tuning procedure used in CIT. Two variants (IT-PAD, IT-ADD) of the transformation function  $g(\cdot, \cdot)$ , showing examples of the newly introduced learnable parameters  $\theta_g$  and of the resulting transformed input  $\tilde{x}$ .

### 4.3 Continually tuning a pre-trained model

We advocate for parsimonious approaches to the continual learning problem. As such, we consider a specific setting in which learning is performed by an edge device with limited computational resources, shipped with a neural network that was pre-trained on an initial task (usually exploiting large-scale data). Without any loss of generality, we focus on image classification problems, so that the input of the network is a  $w \times h$  RGB image. As usual, the network can be considered to be composed by a feature extractor  $m$  that extracts higher-level features, and a classification head  $c$  that returns the predicted confidence  $y$  on a set of classes,

$$y = c(m(x, \theta_m), \theta_c),$$

where  $\theta_m$  and  $\theta_c$  are the parameters (weights and biases) of the feature extractor and the classification head, respectively, and  $y$  is a vector with a number of components equal to the number of classes. For simplicity, we consider the classification head to be composed only of the last linear-projection and non-linearity. The embedded knowledge can be transferred to other somewhat related tasks by removing  $c$  and replacing it with a task-specific head  $\hat{c}$ , with its own new  $\theta_{\hat{c}}$ . The original  $m$  acts as a backbone, and  $\theta_m$  can be fine-tuned on the new task, together with learning from scratch the new  $\theta_{\hat{c}}$ . In this chapter, we indicate with FT<sup>1</sup> such a fine-tuning

<sup>1</sup>FT stands for *Fine-Tuning*, and it is not related to *Friendly Training* of Ch. 3.



approach, that can be possibly restricted to  $\theta_{\hat{c}}$  and a subset of  $\theta_m$ . We refer to that setting as FT-Partial. We further distinguish these cases from a more lightweight option called Bias Tuning (BT), which targets what is known to be a particularly small and expressive subset of model parameters (Cai et al., 2020), i.e., the biases of the neurons of the whole network, together with the usual  $\theta_{\hat{c}}$ .

We focus on the popular CL setting in which learning consists of a certain number  $T \geq 1$  of separated training sessions,  $\mathcal{S}_j, j = 1, \dots, T$ , where  $T$  could be potentially infinite (lifelong learning). Each session comes with data  $\mathcal{D}_j$ , and there is no overlap between data batches of different sessions,  $\mathcal{D}_j \cap \mathcal{D}_h = \emptyset, \forall (j, h)$ . In each  $\mathcal{S}_j$  the learner is presented with a task and after the end of the session the task data are no more available for further training. Before starting the CL process, we plug a novel classification head  $\hat{c}$  on top of the pre-trained  $m$ , using a large number of output neurons (at least equal to the expected total number of classes at the end of the whole learning process), and randomly initializing its parameters. Of course, we assume that the pre-training dataset is sufficiently generic to be helpful for tackling a variety of downstream learning problems. The final goal is to effectively tune the model in a progressive real-time manner, using data coming from the stream of tasks, without buffering past information. We indicate with  $\theta_*^{(j)}$  the values of the parameters after session  $\mathcal{S}_j$ , where  $*$  is a placeholder for all the different aforementioned subsets of parameters. We expect the overall model at the end of the sequence of tasks,

$$y = \hat{c} \left( m \left( x, \theta_m^{(T)} \right), \theta_{\hat{c}}^{(T)} \right),$$

to have high average accuracy on all the tasks, keeping the average forgetting of knowledge at minimum.

We will mostly focus on the class-incremental setting with some insights on the domain-incremental one, in both cases in a fully supervised scenario. In the *class-incremental* setting we are given data partitioned into a large number of classes and we implement the continual learning setting by limiting each session  $\mathcal{S}_j$  to a specific subset of them (disjoint subsets). At training time, during session  $\mathcal{S}_j$ , the classifier outputs confidence scores  $y$  limited to the session-related classes, thus exploiting a portion of the head  $\hat{c}$ , referred to as  $\hat{c}_j$ .<sup>2</sup> Gradients are only computed for the parameters that involve the output neurons in  $\hat{c}_j$ , whose values are indicated with  $\theta_{\hat{c}_j}$ . This is an instance of the so-called “label trick” (Zeno et al., 2021), a rather simple technique that has a very effective role in preventing interference and, as such, it is considered as a stable part of all the models. In the *domain-incremental case* the learner is presented with new data labelled over the same set of classes through the sessions, but originating from a different data population, thus the label trick does not apply. In fact, for each  $\mathcal{S}_j$  the classifier outputs the full set of confidence scores

<sup>2</sup>In case of softmax activation on the output units, our notation  $y$  refers to the non-normalized logits (and not to the softmax output).

using the whole  $\hat{c}$ , and gradients are always computed with respect to all the  $\theta_{\hat{c}}$ . In all the CL experiments of this chapter we always assume that the task identity is *not known* at test time, since it is more challenging and more realistic. It is important to remark the difference of what we have described so far from the more common transfer learning process in which all the data ( $\cup_{j=1}^T \mathcal{D}_j$ ) is simultaneously available to tune the model, that we refer to as Joint Learning (JL) setting, since all the new "tasks" are jointly processed. As a matter of fact, learning in a CL setting is significantly more challenging than in JL, given that we expect the model to keep a reasonable performance on past tasks while learning the new ones in a sequential manner, without buffering data across tasks.

### 4.3.1 Proposed approach

We propose to consider a generic input tuning (IT) procedure, in which the original input image  $x$  is transformed into  $\tilde{x}$  by some function  $g$  before being fed to the frozen backbone (Fig. 4.1),

$$\begin{aligned}\tilde{x} &= g(x, \theta_g) \\ y &= \hat{c}(m(\tilde{x}, \theta_m), \theta_{\hat{c}}),\end{aligned}$$

where  $\theta_g$  are *newly introduced* learnable parameters involved in the transformation function only. The purpose of the optimization carried out during the learning process is to jointly train the parameters  $\theta_{\hat{c}}$  of the classifier  $\hat{c}$  together with the novel input tuning parameters  $\theta_g$ . The first input tuning approach we consider, which we refer to as IT-PAD, consists in framing the input image with a border (referred to as *Frame*) of learnable "pixels", described by  $\theta_g$ . Another simple alternative, henceforth denoted as IT-ADD, consists in transforming the input by adding to  $x$  a learnable tensor  $\theta_g$  (termed as *Perturbation*) of the same shape as the input image, shared by all the possible inputs. Fig. 4.1 shows a visual sketch of these transformations, including an example taken from the experiments (Sec. 4.4).

In principle, this IT tuning method can be used both in Joint Learning (standard supervised learning setting) and in Continual Learning, the setting of our primary interest here. In order to provide a glimpse on the outcome of Continual Input Tuning, we report in Tab. 4.1 the change in accuracy we get when comparing a non-tuned baseline model (i.e., a model in which only  $\theta_{\hat{c}}$  are trained while the whole backbone is kept fixed) with the previously introduced fine-tuning or input tuning procedures, both when performing transfer learning using all the data (JL, third column), and when performing sequential CL (fourth column).

Sequential CL is the focus of this chapter and is the setting that will be thoroughly discussed in the following. It is evident that what works very well in the JL setting is not necessarily well-suited for the CL case, confirming the importance of

investigating alternative ways of exploiting pre-trained backbones in CL. Interestingly, the IT instances (IT-PAD and IT-ADD) are the ones that better perform in CL, even if they learn a relatively small set of parameters. From now on, whenever we apply Input Tuning transformation in a CL context, we will use the acronym CIT (Continual Input Tuning).

Tuning	Learnt Parameters	Joint	Continual
None	$\theta_{\hat{c}}$	66.12	44.49
BT	$\theta_{\hat{c}}$ and Biases of $m$ ( $\sim 1k$ )	+4.44	-2.77
FT-Partial1	$\theta_{\hat{c}}$ and $\theta'_m$ (4.7M)	+1.69	-30.47
FT-Partial2	$\theta_{\hat{c}}$ and $\theta''_m$ (8.4M)	-2.16	-32.20
IT-PAD	$\theta_{\hat{c}}$ and <i>Frame</i> (0.1M)	+1.86	<b>+7.09</b>
IT-ADD	$\theta_{\hat{c}}$ and <i>Perturbation</i> (0.15M)	-0.56	+0.32

Table 4.1: Preview of the impact of different tuning approaches on CIFAR100 in the Joint (accuracy) and Continual (average task accuracy measured at the end of the learning sequence) Learning settings. The first row reports the absolute results of the baseline. The *differences* with respect to them are reported in the other rows. Bias Tuning (BT) is very effective when all the examples are simultaneously available for training. On the other hand, IT is a competitive approach in the CL setting, henceforth indicated with Continual Input Tuning (CIT). In that context, partial fine-tuning (FT-Partial) badly fails.  $\theta'_m$  and  $\theta''_m$  are two different subsets of the backbone parameters – see Sec. 4.4 for more details. From now on, whenever we apply Input Tuning transformation in a CL context, we will use the acronym CIT (Continual Input Tuning).

In the case of a continual learning problems spanning clearly heterogeneous distributions (for example in the case of data collected from multiple sources), we can argue that sharing the exact same transformation  $g(\cdot, \theta_g)$  for all the tasks (referred to as *standard approach*, depicted in Fig. 4.2a) may not be optimal. For this reason, we propose to learn a different input transformation for each training session, by learning independent  $\theta_g$ 's. We will indicate with  $\theta_{g_j}$  the transformation parameters learnt in session  $\mathcal{S}_j$ , that are about the  $j$ -th task. In the challenging setting we consider, the task identity is not known at test time, and we are going to solve the task identification task with a heuristic approach. After the training session  $\mathcal{S}_t$ , we transform a test sample  $x$  according all the known transformations  $g(\cdot, \theta_{g_j}^{(j)})$ . Then, we compute representations through the frozen backbone  $m(\cdot, \theta_m)$  and the classification heads. Finally, we aggregate the partial output vectors with concatenation or element-wise maximum if the learning problem is class-incremental or domain-incremental, respectively—see the inference algorithm (Alg. 3 - 4, lines 3 and 5).

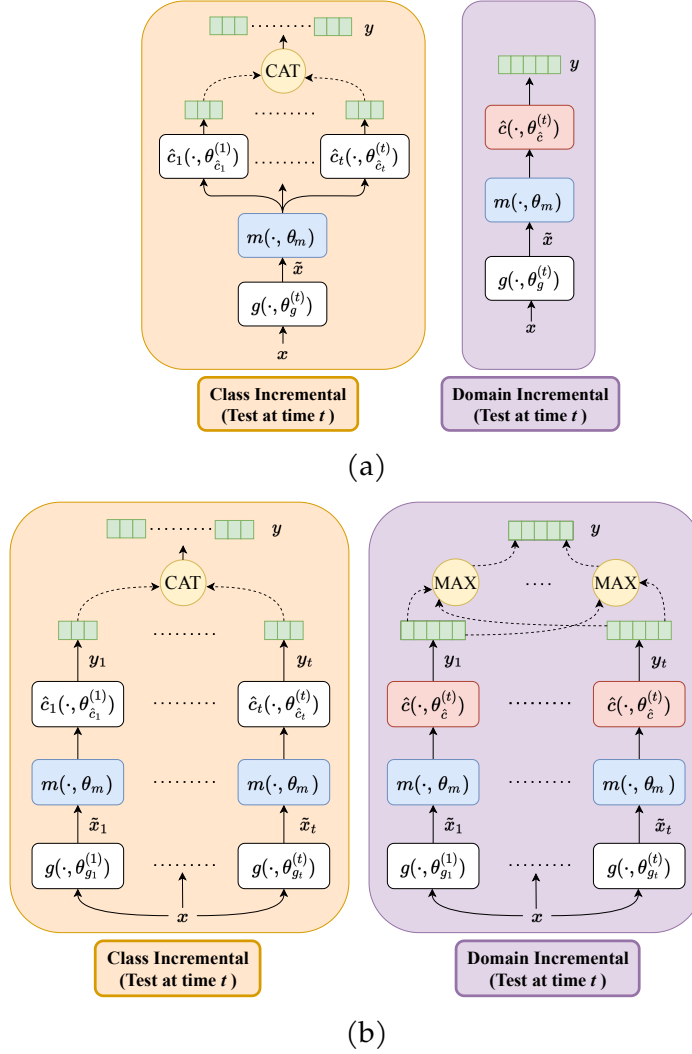


Figure 4.2: Sketch of Continual Input Tuning computational pipeline for the *class-incremental* and *domain-incremental* cases, going from input  $x$  to class-confidence scores  $y$  (logits in the case of softmax) at inference time: (a) standard approach, (b) the *parallel classifier* variant.

Since all these operations can be run in parallel over the different transformations (up to the final  $y_j$ 's), we refer to such classification procedure with the term *parallel classifier* (see Fig. 4.2b).

## 4.4 Experiments

We describe our experimental investigation by introducing the considered datasets, competitors, neural architectures and experimental setup. Subsequently, we will present a quantitative comparison and analyze the results.

---

**Algorithm 3** CIT parallel classifier in class-incremental settings.

---

**Input:** Sample  $x$ , transformations  $\theta_{g_1}^{(1)}, \dots, \theta_{g_t}^{(t)}$ , classification heads  $\theta_{\hat{c}_1}^{(1)}, \dots, \theta_{\hat{c}_t}^{(t)}$

**Output:** Classification vector  $y$ .

```

1: for  $j = 1, \dots, t$  do
2:    $\tilde{x}_j = g(x, \theta_{g_j}^{(j)})$ 
3:    $y_j = \hat{c}_j(m(\tilde{x}_j, \theta_m), \theta_{\hat{c}_j}^{(j)})$ 
4: end for
5:  $y = \text{concat}(\{y_j, j = 1, \dots, t\})$ 
6: return  $y$ 

```

---



---

**Algorithm 4** CIT parallel classifier in domain-incremental settings.

---

**Input:** Sample  $x$ , transformations  $\theta_{g_1}^{(1)}, \dots, \theta_{g_t}^{(t)}$ , classification head  $\theta_{\hat{c}}^{(t)}$

**Output:** Classification vector  $y$ .

```

1: for  $j = 1, \dots, t$  do
2:    $\tilde{x}_j = g(x, \theta_{g_j}^{(j)})$ 
3:    $y_j = \hat{c}(m(\tilde{x}_j, \theta_m), \theta_{\hat{c}}^{(t)})$ 
4: end for
5:  $y = \text{elementwise\_max}(\{y_j, j = 1, \dots, t\})$ 
6: return  $y$ 

```

---

### 4.4.1 Datasets

In order to assess the performance of the proposed learning algorithm, we exploit multiple datasets available in the literature. CIFAR100 (Krizhevsky, 2009) is a popular Image Classification dataset, consisting of 60k  $32 \times 32$  color images from 100 different classes (500 training images per class). RESISC45 dataset (Cheng et al., 2017) is a benchmark for Remote Sensing Image Scene Classification (RESISC). This dataset contains 32k color images, covering 45 scene classes (560 training images for class). FIVEDS (Ebrahimi et al., 2020) is the concatenation of five classic image classification datasets: CIFAR-10 (Krizhevsky, 2009), MNIST (LeCun, 1998), Fashion-MNIST (Xiao et al., 2017), SVHN (Netzer et al., 2011) and notMNIST (Bulatov, 2011). Although each single benchmark is individually fairly easy when using pretrained models, FIVEDS is challenging because of the forgetting arising from very different distributions in the input space. DomainNet (Peng et al., 2019) is a collection of images labelled over 345 classes with multiple drastic domain shifts. In this chapter, we exploit 4 subsets (sketch, real, painting, clipart), totalling 250k training samples and 110k test samples.

We define four different CL problems using the just described four datasets: (i.) CIFAR100/T10, where CIFAR100 is randomly split into 10 tasks (each task con-

tains 10 classes) (*ii.*) RESISC45/T9 with 9 tasks (each task contains 5 classes); (*iii.*) FIVEDS/T5, where each task consists of the 10 classes available in each of the sub-datasets populating FIVEDS, and (*iv.*) DOMAINNET/T4, where each task contains new examples of the same set of classes but from a different domain. While (*i.*, *ii.*, *iii.*) are *class-incremental*, (*iv.*) is *domain-incremental*. Moreover, (*i.*, *ii.*) are Single Source Data, while (*iii.*, *iv.*) are Multiple Source Data, in the sense introduced in Sec. 4.3 (i.e., continual learning problems spanning strongly heterogeneous distributions).

#### 4.4.2 Competitors

We compare the proposed CIT-PAD and CIT-ADD with a baseline model that trains only the classification head (parameters  $\theta_{\hat{c}}$ ). Other competitors are BT and FT-Partial (see Sec. 4.3). In CIT-PAD, CIT-ADD and the baseline model, the learning process is driven by a standard supervised loss that involves the data available in each session, i.e.,  $\mathcal{D}_j$ . Differently, in the case of FT-Partial, the loss is augmented with CL regularizers from state-of-the-art approaches.<sup>3</sup> Our goal is to investigate whether CIT is a competitive strategy without changing the loss function and without introducing extra modules. As a matter of fact, a variety of different specific CL techniques have been developed in the last decade (see Sec. 2.2 for a categorization). In this section, we focus on regularization-based methods, since we stated in our requirements that buffering exemplars is excluded, hence automatically ruling out replay methods. In particular, data-based regularization techniques are mostly inspired by knowledge distillation (Hinton et al., 2014). Learning without Forgetting (LwF) (Li and Hoiem, 2017) is the simplest instance and it basically performs distillation on output logits computed on the currently available data, from the model obtained at the end of session  $\mathcal{S}_{t-1}$  to the current model in  $\mathcal{S}_t$ . Clearly, distillation is restricted to the units of the classes learnt up to time  $\mathcal{S}_t$  and is performed to avoid excessive model drift. Alternatively, Learning without Memorizing (LwM) (Dhar et al., 2019) is geared towards attention. Specifically, it consists of an additional loss used to preserve attention maps over different sessions (Masana et al., 2022). Finally, we also consider prior-based regularization approaches that directly target the weights, such as Elastic Weight Consolidation (EWC) (Kirkpatrick et al., 2017). The importance of the weights is estimated through an approximation of the Fisher Information Matrix, and EWC enforces regularity on the learnable parameters between  $\theta_*^{(t-1)}$  and  $\theta_*^{(t)}$  according to such estimated importance. Zenke et al. (2017) proposed an even simpler instance of EWC, since they show that the importance of each parameter to the fulfilment of a learning task can be estimated by accumulat-

<sup>3</sup>As we will show in Sec. 4.5, fine-tuning without any CL regularizers is impractical due to strong forgetting.

ing individual weight changes; such method is called Path Integral in the following (also known as Synaptic Intelligence).

### 4.4.3 Experimental setup

We focus on a specific class of convolutional networks for image classification, the widely popular ResNets (He et al., 2016), following our edge-oriented scenario in which the computational budget is indeed limited. In particular, we will consider the rather small ResNet-18 (11M parameters), pretrained on ImageNet (Russakovsky et al., 2015). Differently from a Transformer-based architecture, it requires smaller memory and it is less computationally demanding, also when compared to relatively lightweight Transformer - such as ViT-B/16, 84M parameters. In the following we consider two different partial finetuning settings, FT-Partial1 and FT-Partial2, focusing on the last module of the considered architecture, which is made of the two last BasicBlocks<sup>4</sup> and represents a remarkable fraction (70% of the total parameters) of the whole architecture. In FT-Partial1 we restrict the tuning operations to the parameters contained in the last BasicBlock (4.7M), and we refer to these parameters with  $\theta'_m$ . In FT-Partial2 we also include the ones in the penultimate BasicBlock (+3.7M), indicating with  $\theta''_m$  the union of these parameters.

In general, we use cross-entropy as classification loss function, and the Adam optimizer is exploited for all the learnable parameters unless otherwise specified; a smaller batch size  $B$  is employed for the smaller datasets ( $B = 16$  for CIFAR100, RESISC45;  $B = 64$  for FIVEDS, DOMAINNET). In all the experiments, non-backbone parameters are randomly initialized, using the same seed for the different competitors, and we report results averaged over 3 runs with different initializations. Following Buzzega et al. (2020), training for multiple epochs decouples the effects of forgetting and underfitting. As such, unless otherwise stated, we select a sufficient number of epochs to obtain a stable configuration of the parameters at the end of each task. Since we assume to have a limited computational budget, hyperparameters are shared across the different learning problems (i.e., not tuned specifically for the dataset at hand). In all the CIT-PAD experiments we learn a 32-pixel thick border. Concerning CL strategies<sup>5</sup>, we adopt parameters suggested by the respective authors and further investigated by Masana et al. (2022): for LwF we set the temperature to 2; for EWC the fusion of the old and new importance weights is done with  $\alpha = 0.5$ ; for LwM we set  $\beta = \gamma = 1.0$ ; for Path Integral we fix the damping parameter to 0.1 as proposed in the original work (see references for further details).

We measure the average accuracy and average forgetting at the end of the learning sequence (i.e.,  $t = T$ ). The average accuracy  $\bar{a}_T$  at the end of the considered task

<sup>4</sup>Please refer to the PyTorch implementation of the ResNet architecture, [https://pytorch.org/vision/0.8/\\_modules/torchvision/models/resnet.html](https://pytorch.org/vision/0.8/_modules/torchvision/models/resnet.html) for further details.

<sup>5</sup>Refer to the original papers for details on the role of these parameters.

sequence is selected as the main metric, as in most of the continual learning literature. Let be  $\tau$  the task/session index,  $a_{t,\tau}$  is the accuracy computed on the test set of task  $\tau$ , with the model obtained after training on task  $t$  (i.e., with parameters  $\theta_*^{(t)}$ ). Average forgetting  $\bar{f}_T$  is also helpful to evaluate the average magnitude of accuracy drops over the sequence. Formally,

$$\bar{a}_t = \frac{1}{t} \sum_{\tau=1}^t a_{t,\tau}, \quad \bar{f}_t = \frac{1}{t-1} \sum_{\tau=1}^{t-1} \max_{\tau' \in \{1, \dots, t-1\}} (a_{\tau',\tau} - a_{t,\tau}).$$

In all the following results, the terms *accuracy* and *forgetting* refer to the aforementioned average values computed with  $t = T$  (final).

#### 4.4.4 Results

The results of our experimental activities are reported in Tab. 4.2 and Tab. 4.3, analyzed in the following.

In the case of CIFAR100/T10 (Single Source Data, Tab. 4.2) the domain shift from one task to another one in the sequence is relatively small and qualitatively all the data share similar visual features (Parisi et al., 2019). The first noteworthy remark is that the baseline, which only learns the classifier  $\hat{c}$  with the label trick, works surprisingly well with respect to fine-tuning paired with well-known CL regularizers. Interestingly enough, bias tuning (BT) shows no practical improvement over the baseline, given the forgetting due to the incremental nature of the learning problem. The same trend can be observed for the partial fine-tuning options (FT-Partial1, FT-Partial2). On the contrary, both the CIT approaches have at least the same accuracy as the baseline, showing that they are an appropriate way to strive for better performance, with slight additional complexity on top of the label trick baseline. While the improvement provided by CIT-ADD is marginal, CIT-PAD significantly improves the test accuracy without being particularly exposed to forgetting compared to the other options. As expected (Sec. 4.3), the *parallel classifier* approach does not help here, due to relative data homogeneity of different tasks (Single Source Data).

In RESISC45/T9 (Single Source Data, Tab. 4.2), one may wonder whether a relatively high semantic affinity between the pre-training domain (ImageNet) and the target tasks is crucial in order to get improvements with respect to the baseline. In fact, RESISC45 is a publicly available dataset of satellite imagery, which features a quite large semantic and perceptual shift with respect to the pre-training domain. Similarly to the previous case, the regularization-based CL strategies are struggling in achieving and maintaining a good performance throughout the task sequence, dropping to accuracy levels that are lower than the baseline. On the other hand, CIT-PAD is again the best performing option and provides an appreciable accuracy increase, while being less affected by forgetting than the other tuning options (excluding FT-Partial1-LwF, characterized by a consistently lower accuracy though).



Tuning	Learnt Parameters	CIFAR100/T10		RESISC45/T9	
		Accuracy $\uparrow$	Forgetting $\downarrow$	Accuracy $\uparrow$	Forgetting $\downarrow$
None	$\theta_{\hat{c}}$	44.49 $\pm$ 0.22	13.92 $\pm$ 1.49	57.14 $\pm$ 0.50	22.51 $\pm$ 1.15
BT	$\theta_{\hat{c}}$ and Biases of $m$ ( $\sim$ 1k)	41.72 $\pm$ 0.30	29.91 $\pm$ 0.81	46.79 $\pm$ 2.06	35.24 $\pm$ 1.19
FT-Partial1-LwF	$\theta_{\hat{c}}$ and $\theta'_m$ (4.7M)	39.18 $\pm$ 0.83	24.62 $\pm$ 0.65	54.46 $\pm$ 1.15	12.07 $\pm$ 1.41
FT-Partial1-LwM	$\theta_{\hat{c}}$ and $\theta'_m$ (4.7M)	38.26 $\pm$ 0.72	22.19 $\pm$ 0.81	52.51 $\pm$ 1.06	25.92 $\pm$ 1.49
FT-Partial1-EWC	$\theta_{\hat{c}}$ and $\theta'_m$ (4.7M)	39.62 $\pm$ 0.48	22.85 $\pm$ 0.73	54.17 $\pm$ 0.45	25.75 $\pm$ 1.46
FT-Partial1-PathInt	$\theta_{\hat{c}}$ and $\theta'_m$ (4.7M)	37.46 $\pm$ 1.22	23.49 $\pm$ 0.97	52.95 $\pm$ 1.64	24.47 $\pm$ 1.35
FT-Partial2-LwF	$\theta_{\hat{c}}$ and $\theta''_m$ (8.4M)	43.03 $\pm$ 0.49	28.61 $\pm$ 0.22	53.78 $\pm$ 0.45	29.27 $\pm$ 2.05
FT-Partial2-LwM	$\theta_{\hat{c}}$ and $\theta''_m$ (8.4M)	42.06 $\pm$ 0.63	28.57 $\pm$ 1.04	53.94 $\pm$ 0.88	28.94 $\pm$ 2.60
FT-Partial2-EWC	$\theta_{\hat{c}}$ and $\theta''_m$ (8.4M)	41.91 $\pm$ 0.81	26.76 $\pm$ 0.64	56.03 $\pm$ 1.39	27.17 $\pm$ 0.41
FT-Partial2-PathInt	$\theta_{\hat{c}}$ and $\theta''_m$ (8.4M)	42.11 $\pm$ 1.08	27.27 $\pm$ 1.01	54.98 $\pm$ 1.86	26.02 $\pm$ 0.80
CIT-PAD (Standard)	$\theta_{\hat{c}}$ and <i>Frame</i> (0.1M)	<b>51.58</b> $\pm$ 1.78	19.31 $\pm$ 1.46	<b>61.06</b> $\pm$ 0.99	16.99 $\pm$ 0.94
CIT-ADD (Standard)	$\theta_{\hat{c}}$ and <i>Perturbation</i> (0.15M)	44.81 $\pm$ 0.83	21.01 $\pm$ 0.85	57.18 $\pm$ 1.63	19.43 $\pm$ 0.64
CIT-PAD (Parallel)	$\theta_{\hat{c}}$ and <i>Frame</i> (0.1M/task)	46.54 $\pm$ 0.82	12.88 $\pm$ 2.27	55.84 $\pm$ 1.59	19.68 $\pm$ 1.81
CIT-ADD (Parallel)	$\theta_{\hat{c}}$ and <i>Perturb.</i> (0.15M/task)	41.82 $\pm$ 1.23	13.05 $\pm$ 1.14	52.47 $\pm$ 2.83	18.29 $\pm$ 2.51

Table 4.2: Average accuracy ( $\uparrow$ , higher is better) and forgetting ( $\downarrow$ , lower is better) measured at the end of the learning sequence on Single Source Data datasets. In the second column we report the set of parameters subject to optimization: we always learn the classification head; we tune network weights in the BT, FT-Partial approaches and we learn transformation parameters (*Perturbation* or *Frame*) in the CIT approaches. We report the number of such learnt parameters in brackets.

In FIVEDS/T5 (Multiple Source Data, Tab. 4.3) we experiment with data coming from multiple sources, possibly distant in the semantic and perceptual spaces. Interestingly enough, in this case LwF provides the highest performance. Given that it has been originally proposed as a task-incremental strategy it is not surprising that it performs well on a sequence of very distinct tasks (both in the semantic and in the perceptual spaces). At the same time, we can see that CIT-PAD and CIT-ADD provide valuable improvement when implemented in the *parallel classifier*. Moreover, compared to LwF and Path Integral, the amount of learnt parameters (and the computational burden) is smaller, there is no need to store tensors of the same size as the weights (importance weights for PathInt, model snapshot for LwF) and forgetting is lower.

The case of DOMAINNET/T4 (Multiple Source Data, Tab. 4.3) departs from the previous ones, being it a domain-incremental setting. The semantic space is shared by all the tasks, which feature remarkably different visual styles and perceptual features (color, texture, etc.). Although the obtained accuracy may seem a bit low, it is important to remark that it is a very challenging learning problem, with many examples that are hard even for humans and especially for convolutional networks, that are known to heavily rely on texture (Geirhos et al., 2019). We did not apply LwF and LwM, that do not fit well the domain-incremental setting. In fact, the knowl-

Tuning	Learnt Parameters	FIVEDS/T5		DOMAINNET/T4	
		Accuracy $\uparrow$	Forgetting $\downarrow$	Accuracy $\uparrow$	Forgetting $\downarrow$
None	$\theta_{\hat{c}}$	44.20 $\pm$ 2.79	17.78 $\pm$ 1.30	35.93 $\pm$ 0.13	18.98 $\pm$ 0.32
BT	$\theta_{\hat{c}}$ and Biases of $m$ ( $\sim$ 1k)	20.36 $\pm$ 2.86	56.92 $\pm$ 2.11	38.85 $\pm$ 1.93	23.28 $\pm$ 2.31
FT-Partial1-LwF	$\theta_{\hat{c}}$ and $\theta'_m$ (4.7M)	52.72 $\pm$ 1.15	28.85 $\pm$ 1.70	n.a.	n.a.
FT-Partial1-LwM	$\theta_{\hat{c}}$ and $\theta'_m$ (4.7M)	29.34 $\pm$ 0.85	67.02 $\pm$ 4.05	n.a.	n.a.
FT-Partial1-EWC	$\theta_{\hat{c}}$ and $\theta'_m$ (4.7M)	38.45 $\pm$ 2.02	55.18 $\pm$ 0.76	19.05 $\pm$ 1.77	9.72 $\pm$ 0.71
FT-Partial1-PathInt	$\theta_{\hat{c}}$ and $\theta'_m$ (4.7M)	51.38 $\pm$ 1.34	25.12 $\pm$ 0.52	38.55 $\pm$ 1.48	5.7 $\pm$ 1.86
FT-Partial2-LwF	$\theta_{\hat{c}}$ and $\theta''_m$ (8.4M)	<b>61.45</b> $\pm$ 3.02	28.68 $\pm$ 0.23	n.a.	n.a.
FT-Partial2-LwM	$\theta_{\hat{c}}$ and $\theta''_m$ (8.4M)	40.98 $\pm$ 0.87	35.71 $\pm$ 2.45	n.a.	n.a.
FT-Partial2-EWC	$\theta_{\hat{c}}$ and $\theta''_m$ (8.4M)	46.25 $\pm$ 0.88	53.11 $\pm$ 1.05	29.59 $\pm$ 1.31	11.21 $\pm$ 1.42
FT-Partial2-PathInt	$\theta_{\hat{c}}$ and $\theta''_m$ (8.4M)	60.24 $\pm$ 2.05	29.09 $\pm$ 2.62	<b>39.41</b> $\pm$ 1.65	16.17 $\pm$ 3.68
CIT-PAD (Standard)	$\theta_{\hat{c}}$ and <i>Frame</i> (0.1M)	44.65 $\pm$ 1.23	31.86 $\pm$ 0.89	38.74 $\pm$ 0.44	18.85 $\pm$ 1.39
CIT-ADD (Standard)	$\theta_{\hat{c}}$ and <i>Perturbation</i> (0.15M)	36.09 $\pm$ 1.85	44.29 $\pm$ 0.97	33.67 $\pm$ 0.30	22.45 $\pm$ 0.34
CIT-PAD (Parallel)	$\theta_{\hat{c}}$ and <i>Frame</i> (0.1M/task)	53.36 $\pm$ 0.55	19.97 $\pm$ 1.40	<b>43.93</b> $\pm$ 1.22	16.12 $\pm$ 0.91
CIT-ADD (Parallel)	$\theta_{\hat{c}}$ and <i>Perturb.</i> (0.15M/task)	<b>56.32</b> $\pm$ 0.43	17.75 $\pm$ 0.37	41.18 $\pm$ 0.83	17.34 $\pm$ 1.61

Table 4.3: Average accuracy ( $\uparrow$ , higher is better) and forgetting ( $\downarrow$ , lower is better) measured at the end of the learning sequence on Multiple Source Data datasets. Notice that some CL strategies are not well-suited for the domain-incremental setting (see the main text; invalid configurations are then marked with “n.a.”). In the second column we report the set of parameters subject to optimization: we always learn the classification head; we tune network weights in the BT, FT-Partial approaches and we learn transformation parameters (*Perturbation* or *Frame*) in the CIT approaches. We report the number of such learnt parameters in brackets.

edge distillation term would call for the network output with new data to (a) fit the classification loss and (b) be similar to the output of the model learnt at the previous task, which is not expected to help in effectively learning with low forgetting. Also, it should be noted that in this case the baseline itself cannot exploit the label trick, given that each task contains data for the entire class set. As such, several methods offer improvements over the baseline, including the quite simple bias tuning (BT). Moreover, Path Integral gives also a similar improvement, although necessitating to compute and store parameter-specific importance weights. On the other hand, it can be clearly appreciated that the proposed CIT-PAD implemented with *parallel classifier* features the highest accuracy, confirming the importance of learning independent transformations in Multiple Source Data.

## 4.5 Discussion

From what emerged in Sec. 4.4.4, CIT has a positive effect on the average accuracy, especially for CIT-PAD that always increases the accuracy with respect to the baseline, both in class-incremental and domain-incremental settings. In the case of Multiple

Source Data, we can get consistently better performance with the *parallel* variant. We perform additional experiments aimed at gaining more insights on the previously analyzed results, focusing on the CIFAR100/T10 learning problem.

In Fig. 4.3 we report the accuracy obtained in a comparative experiment with 5 different variants of the CIT-PAD scheme. (i.) CIT-PAD-Online is the setting in which training is performed with a single pass on the training data. This speeds up learning but it greatly reduces the extent of the improvement at the end of training (compared to Tab. 4.2). (ii.) CIT-PAD-Fix refers to the setting in which we kept fixed the *Frame* (the learnable padding) after the first task. Results show that learning the additional pixels is really beneficial only if they are allowed to adapt to the slight variations of the different tasks. (iii.) CIT-PAD-Small is obtained reducing by a factor of 4 the thickness of the frame border, decreasing the total amount of additional parameters by a factor of 5; it is interesting to notice that the accuracy is 4% higher than the Baseline (first row of Tab. 4.2) also in that setting. (iv.) CIT-PAD-Latent is about applying the padding operation in a latent space (right after the first two convolutional layers) and benefits of a comparable improvement with a similar amount of additional learnable parameters ( $< 0.1M$ ). On the other hand, (v.) combining the most promising IT approach with BT (CIT-PAD + Bias), completely vanishes any improvement, coherently with observations in (Jia et al., 2022). This analysis confirms the value of the simple-but-effective vanilla CIT-PAD scheme.

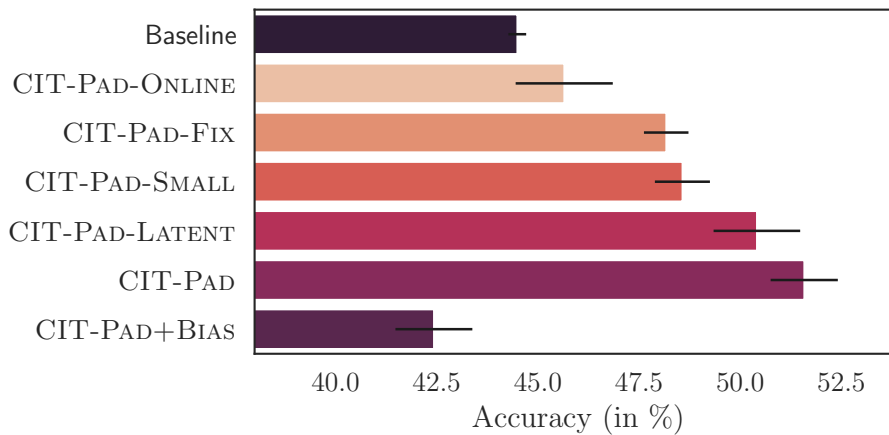


Figure 4.3: Average test accuracy measured at the end of the learning sequence, for variants of CIT-PAD in CIFAR100/T10. Performing the transformation in some latent space (instead of the input) is under-performing and adding bias tuning seems to be counterproductive.

In Tab. 4.4, we further investigate the online setting, showing the test accuracy for the different methods. In general, the gap between the CIT-PAD approach and all the competitors is even wider with respect to the multi-epoch setting, given that fine-tuning a large portion of the network would require a larger amount of update

steps in order to obtain a stable configuration of the weights, conflicting with the single-epoch constraint.

Tuning	Learnt Parameters	Accuracy $\uparrow$	Forgetting $\downarrow$
None	$\theta_{\hat{c}}$	41.38 $\pm 0.56$	14.11 $\pm 1.88$
BT	$\theta_{\hat{c}}$ and Biases ( $\sim 1k$ )	43.94 $\pm 2.31$	24.87 $\pm 3.18$
FT-Partial1 (LwF)	$\theta_{\hat{c}}$ and $\theta'_m$ (4.7M)	26.07 $\pm 0.41$	20.19 $\pm 0.92$
FT-Partial1 (LwM)	$\theta_{\hat{c}}$ and $\theta'_m$ (4.7M)	26.09 $\pm 0.64$	20.08 $\pm 0.68$
FT-Partial1 (EWC)	$\theta_{\hat{c}}$ and $\theta'_m$ (4.7M)	25.88 $\pm 0.83$	23.11 $\pm 0.77$
FT-Partial1 (PathInt)	$\theta_{\hat{c}}$ and $\theta'_m$ (4.7M)	26.06 $\pm 1.98$	20.84 $\pm 1.41$
FT-Partial2 (LwF)	$\theta_{\hat{c}}$ and $\theta''_m$ (8.4M)	29.13 $\pm 0.71$	22.48 $\pm 0.86$
FT-Partial2 (LwM)	$\theta_{\hat{c}}$ and $\theta''_m$ (8.4M)	22.84 $\pm 0.55$	31.83 $\pm 1.29$
FT-Partial2 (EWC)	$\theta_{\hat{c}}$ and $\theta''_m$ (8.4M)	23.14 $\pm 0.32$	21.32 $\pm 0.58$
FT-Partial2 (PathInt)	$\theta_{\hat{c}}$ and $\theta''_m$ (8.4M)	26.69 $\pm 1.72$	24.85 $\pm 1.95$
CIT-PAD (Standard)	$\theta_{\hat{c}}$ and <i>Frame</i> (0.1M)	<b>45.65</b> $\pm 1.20$	17.89 $\pm 1.92$
CIT-ADD (Standard)	$\theta_{\hat{c}}$ and <i>Perturb.</i> (0.15M)	42.48 $\pm 1.34$	14.87 $\pm 0.82$

Table 4.4: Results on CIFAR100/T10 in the online setting: average accuracy and forgetting are measured at the end of the learning sequence. Gap between CIT-PAD and the competitors is even wider than the one detected in the multi-epoch setting (Tab. 4.2).

In Fig. 4.4 we provide some insights on the test accuracy, measured during the learning sequence and at the end of it, respectively. In Fig. 4.4 (left), we show that the CIT-PAD approach typically has the highest average accuracy throughout the learning sequence, while using FT-Partial without any CL regularizer is not a viable option. In Fig. 4.4 (right) we can see that the task accuracy of CIT-PAD is pretty uniform over the entire set (with no drastic forgetting on the old tasks) and generally the highest among the considered approaches.

### 4.5.1 Remarks

To summarize, in this chapter we presented Continual Input Tuning, a novel tuning procedure for the exploitation of pre-trained models in the context of continual learning by tweaking the input data (input tuning). We empirically showed that the proposed method is simple but effective in consistently improving the final average accuracy on multiple learning problems—especially when inserting a frame of learnable pixels (CIT-PAD). We also showed that Continual Input Tuning can be promptly extended to the Multiple Source Data case, where multiple learnable transformations increase the adaptation of the model to very heterogeneous data distributions.

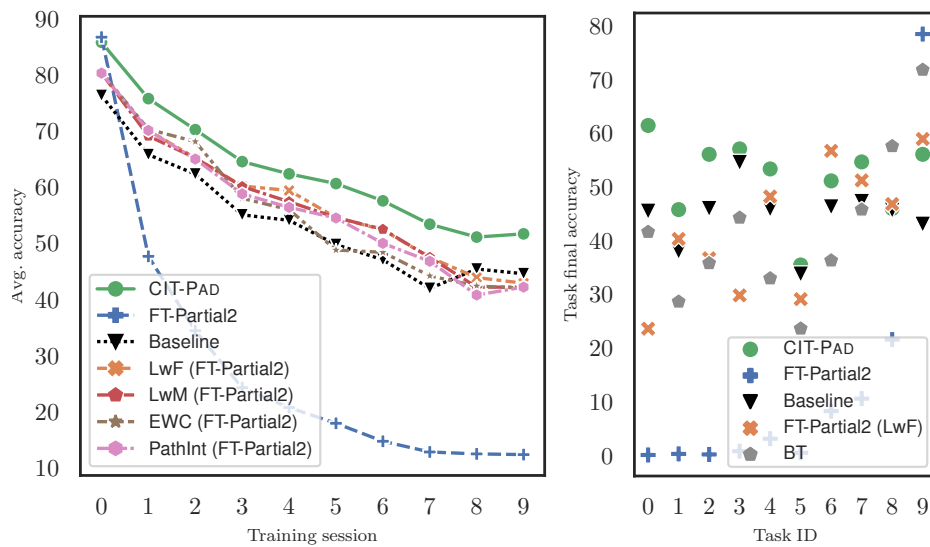


Figure 4.4: CIFAR100/T10, models from Tab. 4.2. Left: Average accuracy during the entire learning sequence. The blue curve (FT-Partial2, with no CL regularizers) highlights the fact that naively fine-tuning is not a practical option. CIT-PAD (green) shows the best behavior throughout all the learning sequence. Right: Task-specific accuracy, at the end of the learning sequence. Fine-tuning without CL regularizers (FT-Partial2, blue) has extremely low accuracy, excluding the very last task. BT (grey) and FT-Partial2 (LwF, orange) best-performing tasks are concentrated in the last part of the sequence (task id  $\geq 6$ ). CIT-PAD consistently beats the baseline and is the best for most of the tasks.

## **Part II**

# **Growing visual agents from video streams**

## Growing visual agents from video streams

In this section of the thesis, we depart from the traditional supervised learning perspective of the first part; our objective now is to develop autonomous visual agents that can learn from their environment. We face this longstanding goal of AI with neural models exposed to continuous observation of video streams, without human intervention. We also emphasize the importance of learning online, thereby eliminating the need to store large volumes of visual data, and addressing concerns related to privacy and data centralization as commented in the Introduction.

In Chapter 5, we explore the feasibility of learning optical flow in dynamic environments without the necessity of storing visual data, adhering to a purely unsupervised online learning approach. Remarkably, we find that optical flow exhibits resilience against forgetting, and we implement straightforward strategies to enhance performance in particularly challenging scenarios. This assessment is preliminary to practical exploitation of motion in Chapter 6.

Chapter 6 introduces two distinct self-supervised learning techniques, driven by a common motivation yet founded upon distinct approaches.

The first initial approach is based on the concept of attention trajectory, inspired by human visual attention in the context of free visual exploration. We employ an attention simulation mechanism which enables the agent to establish semantic connections among pixels traversed by the attention trajectory. These connections stimulate feature development through similarity objectives. Additionally, optical flow is used to distinguish moving objects from the background, providing the ground for more similarity/dissimilarity objectives.

The second approach is more directly based on motion, which is the outcome of a progressive and autonomous learning process, occurring at various levels of the feature hierarchy. Multiple motion flows are estimated with neural networks and characterized by different levels of abstractions, spanning from traditional optical flow to other latent signals originating from higher-level features, hence referred to as higher-level motions. Continuously learning to develop consistent multi-level flows and representations is prone to trivial solutions, which we counteract by introducing a self-supervised contrastive loss, based on flow-induced similarity.

In evaluating the performance of our methods, we operate within an open-set class-incremental setting, a challenging scenario where the classifier must compute

---

membership scores for newly encountered classes upon receiving supervision for the first time. Our visual agent learns to discern and classify objects in real-time. We assess our model on photorealistic synthetic streams and real-worlds videos, comparing to pre-trained state-of-the art feature extractors and to recent unsupervised learning models. Quantitative evaluations confirm substantial performance improvements compared to our earlier approach, representing significant progress in the quest for autonomous visual skill development.



## Chapter 5

# Optical flow estimation with online learning

In the last few years there has been a growing interest in approaches that allow neural networks to learn how to predict optical flow, both in a supervised and, more recently, unsupervised manner. While this clearly opens up the possibility of learning to estimate optical flow in a truly lifelong setting, by processing a potentially endless video stream, existing techniques assume to have access to large datasets and they perform stochastic mini-batch-based gradient optimization, paired with further ad-hoc components. We present an extensive study on how neural networks can learn to estimate optical flow in a continual manner while observing a long video stream and reacting online to the streamed information without any further data buffering. Our analysis considers important model selection issues that might be easily overlooked at a first glance, comparing different neural architectures and also state-of-the-art models pretrained in an offline manner. To this end, we rely on photo-realistic video streams that we specifically created using 3D virtual environments, as well as on a real-world movie. Our results not only show the feasibility of continual unsupervised learning in optical flow estimation, but also indicate that the learned models, in several situations, are comparable to state-of-the-art offline-pretrained networks. Moreover, we show how common issues in continual learning, such as catastrophic forgetting, do not affect the proposed models in a disruptive manner, given the specific properties of the task at hand. This thorough assessment is preliminary to practical exploitation of motion in Chapter 6, where it will work as the main signal for self-supervised feature extraction.

### 5.1 Introduction

Vision offers a natural playground for investigating learning models that continuously evolve over time, especially when considering data streamed from a single

visual source. Dynamical environments (camera movements, objects appearing in the scene, changes in lightning, etc.) require algorithms that adapt to the newly available information without losing the capability of making good predictions on the already processed one. Amongst a large set of interesting vision tasks, in the last few years there has been a growing interest in learning how to predict the optical flow in a visual scene, i.e., the apparent motion<sup>1</sup> of individual pixels on the image plane (Horn and Schunck, 1981). A reliable estimation of the optical flow enables the exploitation of semantic correspondences between frames, and it can be used as precious information for several computer vision-related applications (Betti et al., 2021a). Besides a variety of classic techniques to estimate optical flow (Horn and Schunck, 1981; Lucas and Kanade, 1981; Brox et al., 2004), it has been shown that neural networks can be effectively trained to this end with outstanding results (Dosovitskiy et al., 2015). More recently, the machine learning community moved from supervised approaches (Dosovitskiy et al., 2015; Teed and Deng, 2020) to unsupervised training (Stone et al., 2021).

However, to the best of our knowledge, the optical flow problem has not been investigated in the Continual Learning perspective (see Sec. 2.2 for an introduction), since common training pipelines leverage large datasets processed in an offline manner via randomly shuffling samples in mini-batches. Stochastic gradient descent is commonly exploited, packing randomly selected mini-batches of data samples and generally iterating for multiple epochs, in which the same data are exploited over and over. When data are streamed over time, or whenever we want to build a machine learning system that progressively learns from newly available data without forgetting acquired skills, learning becomes more challenging, but the overall setting sounds more natural and interesting (Betti et al., 2021b). Researchers in the field of Lifelong/Continual Learning (CL) recently proposed a variety of approaches to try to deal with such a setting. However, while a lot of emphasis has been devoted to supervised classification problems, little has been done in the context of Unsupervised Continual Learning (UCL), that is even more intriguing and realistic, due to the intrinsic cost in providing supervisions (Madaan et al., 2022). Apart from a few ad-hoc designed benchmarks (Lomonaco and Maltoni, 2017), evaluating CL algorithms is generally done by adapting well-known classic datasets for machine learning (De Lange et al., 2021).

In this chapter we propose to frame the problem of learning to estimate optical flow with neural networks in the context of UCL, considering temporally-correlated data continuously streamed from a single source and performing single-pass online learning. We study and evaluate how neural networks behave in this online con-

---

<sup>1</sup>In this chapter we will use the terms *flow* and *motion* interchangeably, with the notion of motion that exclusively refers to the apparent displacement of pixels in the visual field, rather than its physical kinematic counterpart.

text, without buffering the video (with the exception of the previous frame only, that is an input of any optical flow estimation method) and without using any ad-hoc CL-based architectures, somewhat representing the most challenging setting of UCL. We rely on modern 3D virtual environments to generate streams in controlled conditions. In detail, the contributions of this chapter are the following. (i) We face a novel problem in the context of UCL, evaluating whether neural networks can learn to predict motion while “observing” a long video stream. We consider different streams in which we inject specific biases to evaluate the behaviour of the networks in critical conditions, proposing proper tricks of the trade to overcome evident issues. (ii) We define an experimental procedure based on 3D virtual environments to generate continuous streams in controlled conditions (different levels of complexity). We include practical considerations on the validation of the model hyper-parameters, that is a frequently overlooked issue. (iii) We also consider the case of a full-length real-world movie and the case of an artificial stream obtained by concatenating the other ones described so far, thus analyzing how the networks adapt to significantly different data and if they forget about the properties of the older streams. (iv) Even though there are no attempts to set a new state-of-the-art in the field, we compare with existing state-of-the-art models trained in an offline manner, using large collections of data samples.

## 5.2 Relation with existing works

Research contained in this chapter can be framed in the context of online UCL, and it is inspired by works concerning optical flow estimation with neural networks. To our best knowledge, this is the first study that evaluates the feasibility of learning to predict motion in online UCL settings. In the following, we describe some interesting works in CL/UCL and optical flow estimation (without and with neural networks).

**Unsupervised Continual Learning.** In the CL (see Sec. 2.2) scientific community, there is still little research on continual exploitation of new data in the lack of further supervision, and more in general, in unsupervised tasks. In the field of self-supervised learning for visual features, we mention the recent work of Madaan et al. (2022), that adopts standard classification benchmarks for evaluation. A different line in neural networks emphasizes the importance of time when developing agents without supervision. Inspired by the principle of least action in physics, Betti et al. (2020a) proposed a framework that deals in a principled way with all the cases in which data become available over time, that was then evaluated in the maximization of the mutual information on a video stream paired with a human-like attention model (Tiezzi et al., 2020). The problem studied in this chapter is an instance of UCL, and it could be easily framed in such a theory, since it is approached with

an online learning strategy.

**Optical Flow.** Horn and Schunck (1981) were the first to formalize the problem of optical flow estimation as the minimization of a functional, considering brightness invariance and regularization. However, such a method has a number of weaknesses, including those that are due to changes in illumination, occlusions and large displacements. Another popular approach is the one of Lucas and Kanade (1981), who proposed a least-square technique to estimate a locally uniform velocity field, with neighboring pixels that feature the same velocity within small patches. The output is sparse but more robust to outliers. Several authors have extended the initial algorithm (Horn and Schunck, 1981) in the attempt of dealing with its weaknesses, such as Brox et al. (2004), who devised a variational algorithm which features robustness to additive illumination changes and better handling of large displacements thanks to a coarse-to-fine strategy. Experimental comparison with Horn and Schunck’s algorithm is provided in Sec. 5.5.

**Optical Flow Estimation with Neural Networks.** Dosovitskiy et al. (2015) originally demonstrated that neural networks with a specific architecture (FlowNet) can be effectively trained to predict motion in a supervised manner. Besides being more accurate than classical methods, learned models can be faster at inference because all optimization occurs during training and common hardware can be efficiently exploited (e.g., GPUs). Supervision typically comes from synthetic data (real-world labeled data are scarce). Teed and Deng (2020) proposed RAFT, which includes recurrent computations on the so-called cost volume, taking into account all the pairs of pixels of the frame. They emulate the steps of a first-order optimization procedure to compute a single high-resolution flow field, which differs from the prevailing coarse-to-fine design. Recently, unsupervised learning was applied to train neural networks to predict optical flow (Jonschkowski et al., 2020). Unsupervised methods are very attractive since they can leverage abundant collections of videos. Most methods work with the warping paradigm, where the estimated flow field is used to warp back a frame into the previous one, and the network is optimized to reduce the photometric distance loss between the real frame and the reconstructed one. A smoothness penalty, computed on the predicted flow, is typically proposed as regularization. Stone et al. (2021) recently adapted the current best performing supervised model (RAFT) to the unsupervised setting. They proposed SMURF, embracing the self-supervision student-teacher paradigm, trained with extensive data augmentation and occlusion handling. Our work differs from these approaches since, to the best of our knowledge, we are the first to study the optical flow problem in a UCL perspective.

### 5.3 Continual learning of optical flow estimation

Amongst a variety of specific formulations, the problem of estimating optical flow is based on the so-called brightness constancy assumption paired with a spatial regularizer to enforce smoothness in the solution (Brox et al., 2004). In detail, the scalar flow fields  $U^* = \{u_{xy}^*, \forall x, \forall y\}$  (horizontal direction) and  $V^* = \{v_{xy}^*, \forall x, \forall y\}$  (vertical direction) between the image at time  $t - \delta$  and the one at time  $t$ , for a small  $\delta$ , are given by

$$\arg \min_{U, V} L(U, V, t) = \arg \min_{U, V} \iint \rho(I(x + u_{xy}, y + v_{xy}, t) - I(x, y, t - \delta)) + \lambda(\|\nabla u_{xy}\|^2 + \|\nabla v_{xy}\|^2) dx dy, \quad (5.1)$$

being  $x, y$  spatial coordinates and  $I(x, y, t)$  the intensity of a pixel, while  $\rho$  is a penalty function, such as  $\rho(a) = a^2$ . The scalar  $\lambda > 0$  weighs a spatial regularizer, while  $\nabla$  is the spatial gradient operator. Computing  $I(x + u_{xy}, y + v_{xy}, t)$  for all  $(x, y)$ 's basically consists in *warping* the image at time  $t$  according to displacements collected in  $U$  and  $V$ . The formulation of Horn and Schunck (1981) is based on the minimization of a functional that is analogous to the one of Eq. 5.1, although we used the linearized version of the constancy assumption (Brox et al., 2004).

Suppose we are given a video stream defined on the time interval  $[0, T)$ , where  $T$  could be potentially infinite. The previous functional naturally extends to

$$\arg \min_{\bar{U}, \bar{V}} \int_0^T L(U_t, V_t, t) dt, \quad (5.2)$$

where  $\bar{U} = \{U_t, \forall t\}$  and  $\bar{V} = \{V_t, \forall t\}$ ,  $\delta = dt$ , and where we assumed  $t - \delta$  to be equal to 0 if negative. We consider the case in which  $U_t$  and  $V_t$  are predicted by a neural network  $f$  with parameters in  $\theta_t$ . Recent literature (Dosovitskiy et al., 2015; Teed and Deng, 2020; Stone et al., 2021) explored the possibility of predicting the displacement field by observing a pair of consecutive images, thus  $U_t = f_U(I_t, I_{t-\delta}, \theta_t)$ ,  $V_t = f_V(I_t, I_{t-\delta}, \theta_t)$ , where we used the subscripts  $U$  and  $V$  to distinguish among the two portions of the output of network  $f$  and where we used the shorthand  $I_t$  to indicate a  $c$ -channel image yielded by the stream at time  $t$ , slightly overloading the previously introduced notation  $I$ , for simplicity. The problem of Eq. 5.2 can be rewritten as a minimization over parameters  $\bar{\theta} = \{\theta_t, \forall t\}$ , thus replacing  $L(U_t, V_t, t)$  with  $\mathcal{L}(\theta_t, t)$  defined as  $\mathcal{L}(\theta_t, t) = L(f_U(I_t, I_{t-\delta}, \theta_t), f_V(I_t, I_{t-\delta}, \theta_t), t)$ .

Let us suppose that we cast the problem in the discrete case, considering a video stream with frame rate  $\nu$  and  $\delta = \nu^{-1}$ . At each discrete time instant  $t$  (multiple of  $\nu^{-1}$ ) the stream yields frame  $I_t$ , and we assume to have access to the previous frame  $I_{t-\delta}$  as well. In this chapter we explore the classic online learning setting in which

the network parameters  $\theta_{t+\delta}$  depend on the ones at  $t$ , and are obtained by updating the current estimate  $\theta_t$  exploiting the pair of frames available at time  $t$ . This ends up in defining the learning step as

$$\theta_{t+\delta} = \theta_t - \gamma \frac{\partial \mathcal{L}(\theta, t)}{\partial \theta} \Big|_{\theta=\theta_t} \quad (5.3)$$

being  $\gamma > 0$  the selected step size. The learning problem is fully unsupervised. We take inspiration from related work, selecting  $\rho$  to be the generalized Charbonnier photometric distance (Sun et al., 2010), which proved to be suitable for optical flow learning since it downplays the importance of small deviations, i.e.,  $\rho(a) = (a^2 + \epsilon)^\alpha$ . While many other components could be inherited from related literature to augment Eq. 5.1 (e.g., edge-aware smoothness, occlusion-oriented terms, etc.—see Jonschkowski et al. (2020)), we specifically aim at evaluating a minimalist implementation that can be eventually enriched in many ways. We remark that we are proposing an always-learning approach, where we perform inference and a learning step at each time instant, nicely adapting to time-variant dynamical domains. As a consequence, it is relevant to evaluate the quality of the model while learning is still taking place.

### 5.3.1 Update policies

When dealing with potentially lifelong horizons, it sounds pretty natural to consider the possibility of not updating the model parameters at each and every timestep  $t$ . This not only reduces the computational burden of the backward stage, but also avoids the network to be affected from redundant information that is likely to be present in frames that are close in time. Moreover, applying the vanilla update rule of Eq. 5.3 at each  $t$  is likely to bias the network capabilities towards the information contained in the very recent frames, “forgetting” the oldest ones (Parisi et al., 2019). In this context, the notion of “forgetting” might imply the lack of capability of estimating the movements of an object that has not been seen for a long time. It might also be due to inadvertent incorporation of lower-level biases that affects motion estimation (for example, long periods in which there is almost no motion or motion always in the same direction). We propose to decrease the correlation between consecutive updates with four simple policies, as alternatives to updating the weights on every new frame (ALWAYS policy):

- **DECIMATION (DEC)**: update the model every  $n$  frames,  $n > 1$ . This simple rule is basically a way to skip frames in a somewhat uninformed manner, given the unknown properties of the input stream.
- **DIFF (DIFF)**: update the model only if the scene is not static, i.e., if the average  $L_2$  distance over the pixel intensities between the frames  $I_t$  and  $I_{t-1}$  is greater than  $q$ , with  $q > 0$ .

- **FLOWDIVERGENCE (DIV)**: after a warmup stage of  $w$  frames, to let the model develop early prediction skills, update the model only when the predicted motion is significantly changing over time. This avoids the model develop direction biases, discarding redundant information coming from objects that move along the same direction for a long time. If  $\tilde{m}_t = [\tilde{u}_t, \tilde{v}_t]$  is the vector that collects the average of  $U_t$  and the one of  $V_t$ , respectively, we update the model only if  $\frac{\|\tilde{m}_t - \tilde{m}_z\|_2}{\|\tilde{m}_z\|_2} > l$ , with  $l > 0$ , being  $z$  the time when the last model update was performed.
- **FLOWMAGNITUDE (MAG)**: after a warmup stage of  $w$  frames, update the model only if a significant portion of the frame is predicted to be moving. At least a fraction  $r$  of the frame pixels must be predicted to have a displacement vector larger than  $h$  to trigger a model update.

Smart update policies can be crucial in general when considering real-world video stream sources. Of course, while **DEC** and **DIV** are pretty generic, **DIFF** and **MAG** are appropriate at mitigating the negative impact of almost static video segments.

### 5.3.2 Evaluation measures and ground truth

Existing work on optical flow estimation typically rely on supervised benchmarks in order to evaluate the trained models (Dosovitskiy et al., 2015; Teed and Deng, 2020), that is not practical in the case of real-world unsupervised data. In fact, the most accessible measures are the ones based on the photometric similarity, in which the previous frame is compared with the reconstructed frame (i.e., the current one, warped by the predicted  $U$  and  $V$ ). We define our notion of **RECONSTRUCTION ACCURACY** as

$$r_{acc}(U, V, t) = \frac{1}{HW} \sum_{x,y} [\|I(x + u_{xy}, y + v_{xy}, t) - I(x, y, t - \delta)\|_\infty < \tau], \quad (5.4)$$

where we considered the resolution  $W \times H$ , and where  $[\cdot]$  is 1 if the condition in brackets is true, otherwise it is 0 (Iverson bracket). The  $\infty$ -norm has been used to take into account images with  $c > 1$  channels (it vanishes in case of  $c = 1$ ). Such measure can be affected by occlusions, but we deem that negligible if the spatio-temporal sampling rates are sufficiently high.

Whenever modern 3D environments for machine learning (see Sec. 5.2) are used to generate video streams, motion-related facilities might be available (Meloni et al., 2021; Gan et al., 2021). The motion field returned by 3D engines is built from full physical knowledge of the environment. Such information is not fully recoverable just by observing projected 2D views, thus excessively relying on the motion field may be tricky when learning and evaluating optical flow. Moreover, the way pixel

velocities are encoded might depend on internal timings of the 3D engine, making it hard to recover displacements for precise comparisons and frame reconstruction.

Although some of the considered streams are from virtual environments (with motion ground truth), for the sake of generality we propose an unsupervised evaluation criterion that applies to any possible stream. Evaluation focuses on consistency, measured by the (i) RECONSTRUCTION ACCURACY of Eq. 5.4. Moreover, in the case of fixed-camera streams with ground-truth available, we also consider motion detection, where the latter is evaluated by computing what we refer to as (ii) MOTION-F1, that is the F1 score in the 2-class problem of predicting whether a pixel is moving or not, starting from flow predictions and comparing with ground-truth from virtual environment. The last ingredient we consider for model selection purposes is the (iii) SPATIAL REGULARITY given by the second term of Eq. 5.1 (the lower the better). In fact, the optical flow problem typically admits spurious solutions that are very irregular and sparse, although with possibly high reconstruction accuracy. Locally-smooth solutions are much more likely to be useful for downstream applications, and they are associated with low values of the SPATIAL REGULARITY term.

### 5.3.3 Model selection

As performance measure we considered the RECONSTRUCTION ACCURACY ( $r_{acc}$ ), paired with the SPATIAL REGULARITY ( $s_{reg}$ ) of the predictions, that can be computed in every stream without requiring any ground truth. In particular, given a pool  $P$  of models, we identify the best model as the one with the largest  $s_{reg}$ , among those whose  $r_{acc}$  is greater than  $c \cdot \max_{j \in P} r_{acc}(j)$ , being  $c \in (0, 1)$ , i.e., preferring those models that are a bit less accurate but more regular. Fig. 5.1 shows a qualitative example of the predictions of a model selected with the proposed criterion, comparing it with a model selected according to  $r_{acc}$  (not considering  $s_{reg}$  at all).

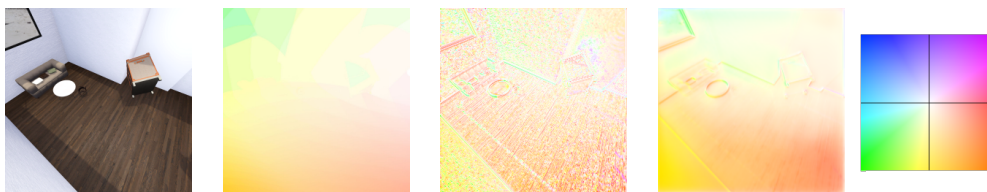


Figure 5.1: From left to right: frame, ground truth, predictions of a model selected with a reconstruction-accuracy-only selection, predictions of a model selected with the proposed criterion. Motion is given by camera rotation, so that accurate solutions should be smooth. Right: Color coding of optical flow vectors. According to this encoding, downward movements will be depicted in yellow and leftward movements in cyan. Static regions will be white.



## 5.4 Experiments

Our experiments have been performed on five different visual streams, referred to as A, B, C, M, and ABCM. The first three ones have been created by ourselves, the fourth one is a real-world movie, while the last one is the concatenation of all the streams. These streams are longer than the ones in popular optical flow benchmarks, and they are designed with increasing level of complexity. We recall that we are not looking for shuffled collections of frame pairs, but for data that naturally evolves over time. We created A, B, C using ThreeDWorld (TDW) (v1.9.1) (Gan et al., 2021), a platform for physical simulation in virtual worlds based on Unity (popular game engine). Manipulating the objects dynamics, we created three photo-realistic living-room-like scenes in which objects move and bounce, and that can provide potentially-endless streams of visual data with dynamics that are not determined by an artificial loop. In detail:

- **STREAM A:** the simplest one, it features translation and rotation of a single object that never leaves the view, similar to FlyingChairs (Dosovitskiy et al., 2015) but with pseudo-realistic rendering.
- **STREAM B:** it features four different objects randomly moving in a realistic scenario.
- **STREAM C:** similar to B, even if the camera is now moving. This stream is much more complex, since it features both very slow movements of objects far from the camera and fast motion patterns from close objects/surfaces.
- **STREAM M:** “1917”, a 2019 British war film directed and produced by Sam Mendes. The film has a duration of 119 minutes (approx. 103 without credits) and it appears as a single long continuous take, without artificial cuts.
- **STREAM ABCM:** the concatenation of all the above streams.

In the case of A, B, and C,<sup>2</sup> we rendered frames at the resolution of  $256 \times 256$  for 1 hour (at 25fps, 90k frames), while M consists of  $\approx 150$ k frames, downsampled at resolution  $320 \times 128$ . See Fig. 5.2 for a showcase of the streams.



Figure 5.2: Stream A, B, C, M considered in this experience. We created from scratch the first three ones, where A and B have fixed camera. Stream M is ©Universal Pictures, DreamWorks Pictures.

<sup>2</sup>Pre-rendered streams and code are available at <https://sailab.diism.unisi.it/continual-unsupervised-learning-for-optical-flow-estimation/>

### 5.4.1 Neural architectures and training

The aim of our experience is to show that reasonable optical flow estimation can be obtained with common neural models through online learning. For this reason, we have evaluated standard convolutional networks, sometimes inheriting their structure from related work (please, refer to the shared code for all the internal details).

The RESUNET network<sup>3</sup> is a U-Net-like architecture (Ronneberger et al., 2015). The contractive backbone is composed by the popular ResNet18 (notice that we removed the Batch Normalization layers). Skip connections are leveraged, repeatedly concatenating current feature maps with lower-level feature maps and applying up-sampling combined with convolution.

The FLOWNETS architecture was proposed by Dosovitskiy et al. (2015), who was the first to demonstrate that standard convolutional networks are capable of solving the optical flow estimation problem as a supervised learning task, relying on large randomly generated synthetic datasets of 3D moving objects. The network consists of two parts: the prediction layers and the refinement layers. Two different approaches were presented. In the first one (FLOWNETS), they stack the two frames as network input. Basically, the prediction part is a convolutional contractive path (where contraction is given by stride), while the refinement module is responsible of upscaling the flow. We stick with this architecture since the other one (FLOWNETC) is more involved and was shown to provide marginal benefits.

ND CONV is a standard convolutional network composed by 8 layers (each one equipped with 5x5 filters) that maintains the same image-resolution throughout the whole architecture, without any downsamplings/poolings. The layers are composed by 32, 64, 64, 128, 128, 64, 64 filters banks, respectively, and ReLU activation functions.

DILND CONV is very similar to the latter model, apart from a dilation factor of 2 on the last four layers (see the provided code repository for further details).

All the models take as input the concatenation of the frames ( $I_{t-1}, I_t$ ) along the channel dimension. We adopted typical parameters for the Charbonnier distance ( $\alpha = 0.5, \epsilon = 0.001$ ), as in (Stone et al., 2021). Concerning model selection, we consider a warmup phase up to  $T_{val} \ll T$ . Then, starting from  $t = T_{val}$ , we measure performance on two consecutive windows, each of them of length  $\beta$ . In the first window the networks are updated, while in the second one they are kept frozen. The idea is to look for a model that maximizes the performance in both the windows, thus able to learn by adapting to the dynamical features of the stream (first window), but also durably embedding meaningful knowledge in the network weights (second window). To rank the models, we use the criterion described in Sec. 5.3, taking into account RECONSTRUCTION ACCURACY and SPATIAL REGULARITY. Assuming

<sup>3</sup>Implementation available at <https://github.com/usuyama/pytorch-unet/>.

$T_{val} = 14$  minutes (21k frames) and  $\beta = 1$  minute, we picked  $c = 0.95$ , and we tuned  $\lambda$  (Eq. 5.1) and the optimization parameters, considering learning rate  $\gamma$  and also adding a weight decay term controlled by  $\zeta$ . Concerning optimizers, we used Adam, AdamW in PyTorch implementations. We selected the best model among:  $\lambda \in \{0.01, 0.05, 0.1, 0.5, 0.5, 1.0, 5.0\}$ ,  $\gamma \in \{5 \cdot 10^{-6}, 10^{-5}, 5 \cdot 10^{-5}, 10^{-4}\}$ ,  $\zeta \in \{0, 0.01\}$ . Concerning update policies, we set  $w = 15$  minutes = 22.5k frames and we compare DIFF with  $q = 0.001$ , MAG with  $r = 0.02$  and  $h = 0.2$ , Div with  $l = 0.05$ .

Networks are randomly initialized and they learn over the whole stream, accumulating running averages of the metrics of interest. This is what we refer to as “learning” setting, different from the one in which we expose the networks to another repetition of the stream keeping “frozen” the model parameters obtained at the end of the “learning” stage. Results on RECONSTRUCTION ACCURACY are based on  $\tau = 0.025$  (i.e., 2.5% of the  $[0, 1]$  range), while for MOTION - F1 a pixel is marked as static if its flow has  $L_\infty$  norm  $< 0.5$  (see Sec. 5.3.2). We avoid reporting standard deviations over multiple runs, since they are very small (we report them in the Appendix A).

We also compared the proposed solution with results obtained from offline pre-trained baselines; we used ‘MPI-Sintel’ (Butler et al., 2012) training weights when available, since that dataset is quite visually similar to streams A, B, C. We used ‘Flying Chairs’ (Dosovitskiy et al., 2015) weights in the case of FLOWNETS.

Table 5.1: RECONSTRUCTION ACCURACY (%) on the considered streams/settings. We report the best result between the model that always updates its parameters (ALWAYS) and the one that uses Div update policy (adding a \* when results from Div is reported). Best results among the models considered in the proposed experience are in bold. For the sake of readability, FROZEN values of REFERENCE models are reported also under LEARNING (such models are pretrained and kept frozen).

Model	LEARNING					FROZEN					
	A	B	C	M	ABCM	A	B	C	M	ABCM	
RESUNET	86.7	<b>76.8</b>	92.3	85.6	87.4	84.0	<b>78.6</b> *	90.0 *	86.1	<b>88.4</b>	
NDCONV	<b>87.4</b>	76.3	92.9	<b>87.3</b>	<b>87.5</b>	86.0	77.5	91.2	<b>87.9</b>	87.7	
DNDCONV	86.0	76.3 *	93.0	86.4	86.9	<b>86.2</b>	77.3 *	91.1	86.5 *	87.3	
FLOWNETS	83.7	73.4 *	<b>93.1</b>	82.7	84.3	81.7 *	69.3 *	<b>91.8</b>	83.3	84.3	
REFERENCE	NULL	65.2	65.9	90.5	74.6	74.1	65.2	65.9	90.5	74.6	74.1
	FLOWNETS	76.2	69.3	91.4	79.5	79.2	76.2	69.3	91.4	79.5	79.2
	RAFT-SMALL	83.9	73.9	94.6	86.3	84.9	83.9	73.9	94.6	86.3	84.9
	SMURF	87.6	79.8	95.8	90.1	88.6	87.6	79.8	95.8	90.1	88.6
	RAFT	85.9	77.2	95.4	88.0	86.9	85.9	77.2	95.4	88.0	86.9

## 5.4.2 Results

In Tab. 5.1 we report results from both our online-trained models (top rows) and, as a bare reference, from popular state-of-the-art offline-pretrained models (bottom rows) for optical flow estimation. For comparison, we also show a dummy predictor (NULL) which always predicts zero flow. It is evident that online-trained models are able to learn to predict optical flow in a satisfactory way, well overcoming the reference NULL predictor and, surprisingly, also some state-of-the-art offline-pretrained models. Pretrained FLOWNETS (Dosovitskiy et al., 2015) works significantly worse than our models, since it is designed and trained mostly on large-displacement flows, while we considered pretty smooth streams. On the other hand, we acknowledge that other offline-pretrained models such as RAFT (Teed and Deng, 2020) and SMURF (Stone et al., 2021) often achieve higher performance. Of course, our models (top rows) start learning from scratch, thus the outcome of the “learning” setting is negatively biased by the inevitably low performance in the early stages of life.<sup>4</sup> Moreover, the reference state-of-the-art exploits several tricks to improve their quality (edge awareness, occlusion detection, etc.), while our models are working in a plain minimalistic setting. Nonetheless, it is interesting to notice that RAFT-SMALL (Teed and Deng, 2020) does not perform largely better. More importantly, Tab. 5.1 shows that the distance between our models and the offline-pretrained ones is further reduced, and sometimes overturned, if we employ a longer stream (ABCM), confirming the capability of improving over longer horizons. When considering the “frozen” setting, we can clearly see that the performance of our models are equivalent or even slightly better than while *learning*, indicating that the networks are not just adapting on-the-fly to the last seen frames. Overall, ND CONV behaved pretty well in all the considered streams. All the policies of Sec. 5.3.1 were evaluated, with only DIV that allowed to improve the always-update case. For this reason, in Tab. 5.1 we marked those cases in which we found such an improvement. Fig. 5.3 reports a qualitative visualization of the motion predicted on a frame from stream M.

The results of Tab. 5.1, “frozen” setting, seem to suggest that no evident catastrophic forgetting is taking place. Differently from many other vision tasks, both semantic ones (like image classification (De Lange et al., 2021)) and perceptual ones (like depth estimation (Zhang et al., 2020b)), in optical flow there is a big chance that new frames can be effectively handled leveraging previously acquired skills, so that interference is reduced and no special methodology is needed.

---

<sup>4</sup>We remark that our models benefit from longer exposure to the data, further reducing the gap with offline-trained state-of-the-art (see Appendix, Tab. A.7, where we repeated each stream twice).



Figure 5.3: Qualitative comparison of different models on a scene from *STREAM M*, where two soldiers are talking while slightly moving their bodies. *SMURF* (offline pre-trained) output is pretty clear but it suppresses small movements (near the cannon two soldiers are moving, one towards the right, the other one is lowering on his knees, only partially detected) and sometimes oversimplifying. The online-trained models are more sensitive to small changes. The output of *ND CONV* and *DN CONV* is quite similar to *RESUNET*, while *FLOWNETS* overemphasizes the slight motion of the man in the foreground and results to be very blurry due to its architecture. See caption of Fig. 5.1 for details on the visualization.

## 5.5 Discussion

However, we are left with the open question on whether the reported averaged measurement is actually the outcome of very unstable predictions over time. In Fig. 5.4, we report the evolution of the *RECONSTRUCTION ACCURACY*, measured on time windows of 1-minute length, when re-watching (*frozen* setting) the streams after having learnt on them. Predictions are quite stable in *A*, *B*, *C*, and more variable in *M*, as expected, since the movie includes significantly larger variability with respect to the other streams. Overall, we do not observe serious performance deterioration in the earliest time windows, the further from the last updates to model weights—see also Tables A.3, A.4 in the Appendix. The positive impact of *Div* criterion is more evident when measuring *MOTION-F1* in those streams where the camera is fixed (in the other streams almost everything is moving, so we will not discuss such metric), as shown in Tab. 5.2. *Div* helps in increasing the discriminative capability in motion detection, especially in the “*frozen*” setting. The different neural models seem to have similar performance in the given task, apart from *FLOWNETS*. Stream *B* is more challenging, with multiple and smaller moving objects, so there might be larger prediction errors. We further inspected this result in Fig. 5.5, where we compared the *MOTION-F1* (*frozen*) computed on the last minute of each stream (on which we expect less forgetting, since it is near to the moment we froze the weights, i.e. the end of learning) and on the first minute of the stream (oldest data, susceptible to

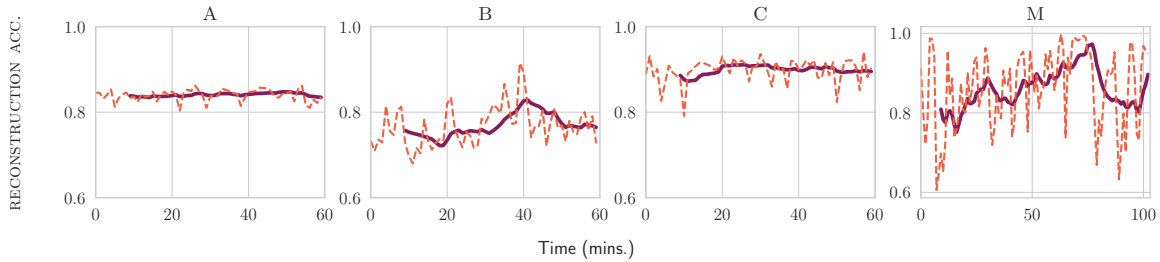


Figure 5.4: RECONSTRUCTION ACCURACY over the whole stream of model RESUNET, computed in small windows of 1-minute length (dashed). The continuous line is a moving average (10-minute interval) of the window-oriented results. Model weights are in the “frozen” setting.

forgetting). The Div strategy can reduce the gap between metrics measured at different time instants (this suggests that the learnt solution is a bit more general). We also show that a simple DEC policy decreases the model performance, at least on medium-length streams as the ones we tested. Overall, some slight forms of performance reduction/forgetting are observed (whenever below the dashed diagonal<sup>5</sup>), but they cannot be classified as catastrophic.

Table 5.2: MOTION-F1 in fixed-camera streams. The \* symbol has the same meaning as in Tab. 5.1.

Model	LEARNING		FROZEN	
	A	B	A	B
RESUNET	0.829	0.661 *	<b>0.834</b> *	0.712 *
NDCONV	<b>0.840</b>	0.667	0.818 *	<b>0.738</b> *
DNDCONV	0.836	<b>0.682</b> *	0.827	0.701 *
FLOWNETS	0.771 *	0.623 *	0.784 *	0.425 *

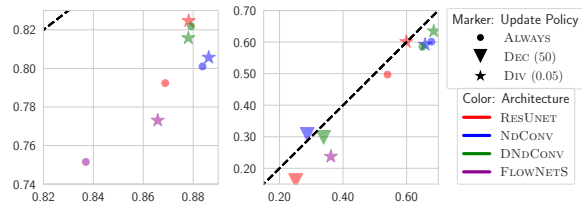


Figure 5.5: MOTION-F1 on recent (x) and older data (y) on stream A (left) and B (right). Div policy (with  $l = 0.05$ ) yields improvements in most of the cases. Some low-performance configurations are out of the scope of the plot.

We conducted a further experience in order to evaluate the sensitivity of the networks in presence of motion-related data biases. We simulated the case in which the scene stands still for a significant amount of time before letting objects move again. This is actually a very likely setting of several real-world scenarios (e.g., surveillance systems, public cameras, etc.). We tested the criteria presented in Sec. 5.3.1 in all our streams, adding 5 minutes of pause after every 5 minutes of playback. Of course, we focused on DIFF and MAG that are specifically designed to handle static shots, and we kept also Div due to its promising behaviour in the previous experiments,

<sup>5</sup>The diagonal, under the assumption of equal difficulty of the first and last minute of the stream, corresponds to the ideal model (no forgetting). From a visual inspection, they appear to be quite similar in terms of difficulty.

together with the vanilla case with no special policies (*ALWAYS*). Fig. 5.6 reports the results in stream A with two networks (similar results are obtained with the other architectures and streams, see the Appendix, Tab. A.5, Tab. A.6). While in the “*learning*” case networks are able to learn and predict in an appropriate way (eventually recovering from biases), moving to the “*frozen*” setting we get a huge performance drop in *ALWAYS* and, partially, in *DIV*, suggesting that the networks have completely embedded the artificially-induced bias. Differently, the *DIFF* and *MAG* criteria significantly help. This experience, when paired with the previous ones, remarks the importance of both *DIFF* (or *MAG*) and *DIV* criteria, to cope with static shots and to filter out other redundant portions of the stream.

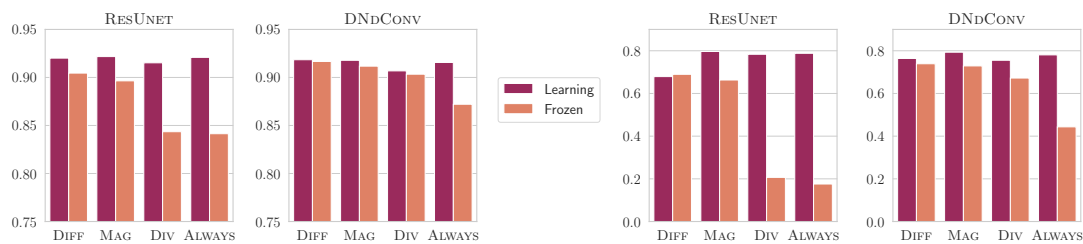


Figure 5.6: RECONSTRUCTION ACCURACY (left) and MOTION-F1 (right) with two sample architectures (stream A, with injected static shots). Sequence-filtering criteria significantly improve the quality of the solutions (frozen weights).

In the spirit of evaluating even longer continual learning settings, we deepen our analysis in stream ABCM (see Tab. A.8, Appendix, for more results). The plots of Fig. 5.7 start in “*learning*” mode, while the rightmost part, after the vertical dotted line, is about the “*frozen*” mode. Together with the already discussed *DIV* criterion, we also considered the not-yet promising *DEC*, as well as a reference criterion that *RESETS* the network weights at the beginning of each sub-stream and learns with the *ALWAYS* policy. The informed *DIV* strategy ( $l = 0.05$ ) confirms its versatility, and *DEC* ( $n = 150$ ) also reduces the gap with the other criteria on the very long run, even though the former is still preferable. There is an evident margin between *RESET* and the best competitors, showing that starting to process a new sub-stream from an already settled configuration (although obtained on different streams) is beneficial—e.g., see sub-stream C. Again, we observe no evident forgetting due to tuning on data from other sub-streams, and the networks benefit from learning on a larger variety of data.

In order to better grasp the benefits of using a neural network to predict optical flow compared to a classic iterative algorithm such as the one of Horn & Shunck (HS), we evaluated our GPU-based implementations of HS (30 or 200 iterations, smoothness coefficient  $\lambda$  validated with the procedure previously described), reporting the MOTION-F1 results in Tab. 5.3 (see Appendix A for more results). Comparison is against our neural models, after having learnt from long streams, achiev-

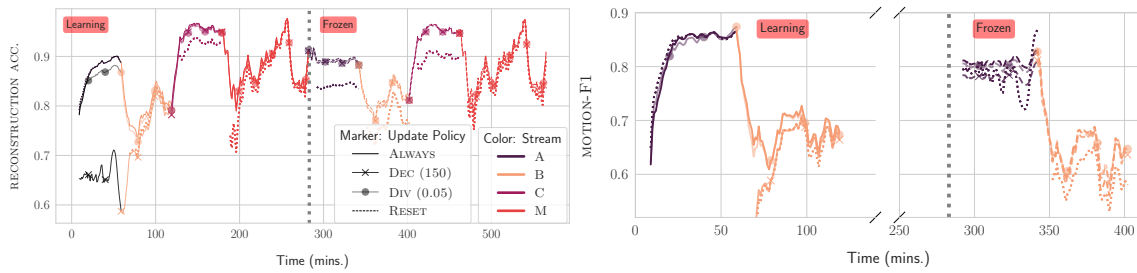


Figure 5.7: Left: RECONSTRUCTION ACCURACY in the concatenated stream ABCM. Right: MOTION-F1 in ABCM, focusing on the time range of sub-streams A and B (where the said metric is well-defined); moving average (10-minute interval). Being exposed to other streams improves the performance, both in “learning” (leftmost part of each plot, before the vertical dotted line) and “frozen” settings (rightmost part).

ing higher-quality flow with respect to the classic HS algorithm. This consideration also holds when a warm start is used in HS (i.e., initializing the flow estimation with flow prediction from the previous time instant). Concerning computational demands, Fig. 5.8 shows that, for example, the RESUNET model is much faster than HS (200 iter.) when predicting the optical flow, while it is of comparable speed when accounting for the learning (backward) phase too. The already discussed benefits of the update policies, such as Div, indicate that the backward step is not always needed, thus saving a significant amount of time and making the neural network-based solutions more attractive.

Table 5.3: MOTION-F1, comparing Horn & Shunck (HS) flow, with (*w.s.*) and without *warm start*, and our neural models, streams A and B. Results of neural models (bottom rows) are from Tab. 5.2.

Model	LEARNING		FROZEN	
	A	B	A	B
HS 30	0.574	0.369	0.574	0.369
HS 30 ( <i>w.s.</i> )	0.649	0.489	0.649	0.489
HS 200	0.462	0.322	0.462	0.322
HS 200 ( <i>w.s.</i> )	0.683	0.462	0.683	0.462
RESUNET	0.829	0.661	0.834	0.712
NDCONV	0.840	0.667	0.818	0.738
DNDCONV	0.836	0.682	0.827	0.701
FLOWNETS	0.771	0.623	0.784	0.425

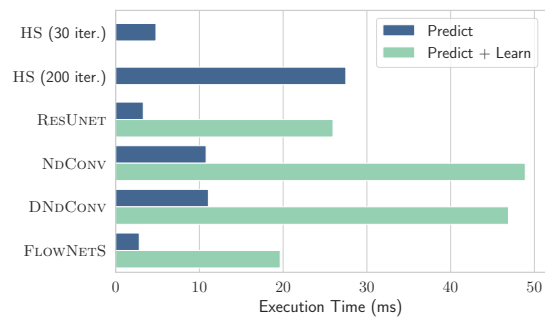


Figure 5.8: Run-times of our implementations (prediction time and prediction+learning/backward time).

### 5.5.1 Remarks

We described a novel experience in the context of Unsupervised Continual Learning, focusing on the problem of learning to predict optical flow by letting a neural



network “watch” a potentially life-long video stream. Results on streams (both from virtual environments and realistic footage) are not far from the ones yielded by advanced state-of-the-art models pre-trained offline. We proposed simple criteria to filter the input data, and we focused on a model selection procedure that copes well with the optical flow prediction problem. In Sec. 6.5, we will leverage this introductory experience as a foundation for crafting a methodology that is based on the concurrent estimation of motion and pixel-level visual representations.

## Chapter 6

# Self-supervised online learning for autonomous visual agents

Devising intelligent agents able to live in an environment and learn by observing the surroundings is a longstanding goal of Artificial Intelligence. From a bare Machine Learning perspective, challenges arise when the agent is prevented from leveraging large fully-annotated dataset, and rather the interactions with supervisory signals are sparsely distributed over space and time. In this chapter, we propose two novel neural-network-based approaches to progressively and autonomously develop pixel-wise representations in a video stream. Such approaches extract information from motion, whether directly or indirectly, whose estimation in a continual learning setting has been investigated in Ch. 5.

The first method, COAT, is based on a human-like attention mechanism that allows the agent to learn by observing what is moving in the attended locations. Spatio-temporal coherence along the attention trajectory, paired with a contrastive term, leads to an unsupervised learning criterion that naturally copes with the considered setting. The second method, CMOSFET, exploits multiple motion-induced constraints, obtaining *motion-conjugated feature representations*. Differently from existing approaches, motion is the outcome of a progressive and autonomous learning process. Multiple motion flows are estimated with neural networks and characterized by different levels of abstractions, spanning from traditional optical flow to other latent signals originating from higher-level features. Continuously learning to develop consistent multi-order flows and representations is prone to trivial solutions, which we counteract by introducing a self-supervised contrastive loss, based on flow-induced similarity.

Differently from most existing works, representations from COAT and CMOSFET are used in open-set class-incremental classification of each frame pixel, relying on few supervisions. Our experiments leverage modern 3D virtual environments and real-worlds videos. We show that the proposed agents can learn to distinguish

objects just by observing the video stream. Moreover, features from state-of-the-art models are not as powerful as one might expect, often outperformed by our methods.

## 6.1 Introduction

In the context of Artificial Intelligence, the idea of designing *agents that exist in an environment and perceive and act* (Russell and Norvig, 2009)<sup>1</sup> is a longstanding goal that introduces a number of challenges.

For instance, it is not trivial to incorporate neural models designed for traditional vision tasks (e.g., image classification) to design a visual agent that learns while watching a video stream, especially when the agent is expected to parsimoniously interact with humans to get information on what it sees. Pretrained models might not always help in capturing properties of entities that belong to the particular environment in which the agent lives (Kornblith et al., 2019), and they are subject to inductive biases. Moreover, the agent should be able to learn synchronously with the continuous video stream, and the target classes are not known in advance. This setting also implies strong correlation in visual information over time, since data cannot be shuffled as commonly done when using stochastic gradient descent.

In order to develop the aforementioned visual agents, the temporal dimension is of utmost importance. Recently, a lot of emphasis has been put on neural models able to learn over time (see Sec. 2.2 for an introduction to continual learning) in the attempt of overcoming the conventional i.i.d. assumption and offline learning (i.e., all training data are instantly available and sampled from a static distribution in an independent manner). Most of the attention has been focused on continual supervised learning, with few notable unsupervised exceptions (Madaan et al., 2022; Rao et al., 2019; Tiezzi et al., 2020; Betti et al., 2022a). Despite the large variety of proposals, learning over time in a continual manner is still a very challenging learning setting, especially when not relying on large buffers of past experiences. Regularization techniques are indeed very powerful, and here we embrace the intuition that motion seems to offer a natural way of regularizing over time.

We advocate for the opportunity of substantially relying on motion to design natural contrastive frameworks (see Sec. 2.3 for an introduction to contrastive learning) for agents that progressively learn from a visual stream. As we introduced in Ch. 1, foundational studies in vision and perception (Spelke, 1990) have shown that the presence of motion significantly enhances the ability of biological perceptual

---

<sup>1</sup>In this chapter, we will use this notion to indicate a *visual* agent, that is a model equipped with tunable parameters and a learning mechanism. The *agency* is then restricted to self-evolution, although in principle could be extended to navigate the environment and produce permanent changes in it (robotics).

systems to identify and segment visual patterns into specific entities. Empirical evidence (Ostrovsky et al., 2009) supports the idea that the capability of parsing static visual scene is systematically achieved way later than the one of parsing dynamic scenes. As a matter of fact, the Gestalt Principles of common fate (Wertheimer, 1938) hypothesized the role of motion as a fundamental cue for visual perception in the early 20th century. Recently, researchers have applied this concept to the domain of Machine Learning for computer vision, leveraging motion-based principles to develop the visual skills of artificial agents. Such intuition has been exploited for designing simple pretext tasks in the context of unsupervised learning (Mahendran et al., 2019), aligning the similarity between pairs of feature vectors to the similarity between the corresponding flow vectors. Pathak et al. (2017) used segments from low-level motion-based grouping to train convolutional networks, yielding easily transferable representations.

We also propose to consider the importance of the notion of focus of attention, which guides agents in wild visual scenes and can be used to attribute precise locations in the human-machine interaction. For example, consider an agent that asks for or receives a specific supervision in a crowded scene, or whenever there is a linguistic interface to exchange information with the human. Without contextualizing the dialogue to what is being precisely observed, the interaction is hardly meaningful: this is where human-machine shared attention can radically improve interaction (see Fig. 6.1). In this context, we are referring to the simulation of *human-like visual attention trajectories* (Zanca et al., 2020), which is different from popular neural attention models (Chaudhari et al., 2019), that are task-oriented and part of the neural computational scheme. In our framework, we assume that supervisions are in the form of a class/instance label about what is being observed, without a precise indication on the boundaries of what is supervised, differently from several Computer Vision tasks (Long et al., 2015).

In this chapter, we consider the case in which the human intervention is rare, and each supervision is about the coordinates of a single pixel with its class/instance label, thus not a signal that can strongly drive the features development. Target classes are not known in advance, and our model includes an open-set approach to abstain from making predictions about unknown entities.

This chapter is organized as follows. Sec. 6.2 briefly points at research areas connected with our proposals. In Sec. 6.3 we describe the learning environment and how we are going to evaluate the outcome of the self-supervised learning process. In Sec. 6.4 we present COAT, the attention-based model, while Sec. 6.5 is dedicated to CMOSFET, the motion-based model. Experiments and results, leveraging virtual environments and real-world videos, are discussed in Sec. 6.6, comparing our results with large offline-trained models.

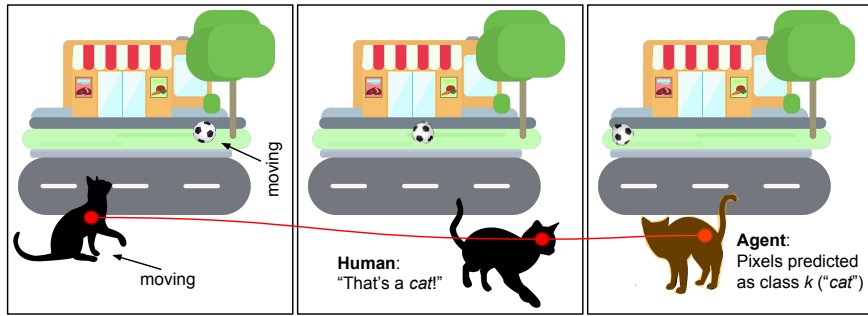


Figure 6.1: Visual stream with static and moving entities. The focus of attention (red circle) disambiguates the subject of the human-machine interaction, making vague supervisions well contextualized (2nd pic). The agent performs self-supervised learning to obtain robust pixel-wise representations in the background, but thanks to this parsimonious supervision procedure, it becomes capable of attaching semantic labels to visual entities.

## 6.2 Relation with existing works

**Online, continual, open-set learning.** An online learning agent progressively learns from a stream of data, continuously adapting to every new processed input instance (Hoi et al., 2018). In the specific case of continual learning (also known as life-long, continuous or incremental learning—Sec. 2.2), the goal of the agent is not fixed a priori but changes over time. In this chapter, the goal is to learn to predict class labels for pixels in a video stream. Such goal is not fully defined in advance, since the agent becomes aware of classes as the human supervisor tells him, and it does not go through a sequence of distinct tasks with clear boundaries—being close to task-free scenario (Aljundi et al., 2019). Open-set classifiers (Scheirer et al., 2012) can distinguish between examples belonging to different training classes but they can also detect whether data do not belong to any known class, that is the case of what we propose. Our work is also class-incremental (Geng et al., 2020), due to the progressive inclusion of new classes after human intervention<sup>2</sup>. With reference to categories in Sec. 2.2, our methods are representation-based (since everything relies on self-supervision) and features are evaluated with template-based classification.

**Self-supervised learning.** In the context of self-supervised learning (Sec. 2.3), contrastive methods are based on the idea of similarity (dissimilarity) between semantically equivalent (distant) inputs, respectively. The vast majority of the methods can be considered *global* contrastive learning, since they are explicitly designed to obtain discriminative features for global tasks (e.g., image classification). Under-

<sup>2</sup>However, it differs from the protocol of the open-world scenario, that goes beyond what we describe in this chapter (Geng et al., 2020). One/few shot supervised models (Min et al., 2021) also learn new classes from few examples, exploiting prior knowledge.

standably, this means that such features are not effective for tasks that require local details (e.g., semantic segmentation); however, this issue is underinvestigated in the literature. Wang et al. (2021) presented a contrastive criterion at pixel level, implemented by correspondence matching (which is, however, fostered by image-level contrastive learning). In this chapter, image-level semantic information is not available, so we will propose pixel-wise criteria that are based on motion. Recently, Xiong et al. (2021) investigated the idea of exploiting optical flow in self-supervised learning, however it is based on offline training (large batches of framepairs with lot of visual variability). In our work, we also exploit optical flow, albeit in a more sophisticated manner. Interestingly, Fini et al. (2022) have used intuitions from knowledge distillation in a self-supervised continual setting, by adding a network that maps the current state of the representations (global) to their past state. As a final note, while self-supervised learning applied to videos typically focuses on global video representation learning (Qian et al., 2021), we, on the contrary, extract pixel-wise representations from individual frames.

**Focus of attention.** Several attempts to model *human-like* focus of attention mechanisms have been presented (Borji, 2019), not only differing in the way they are implemented, but also in the nature of the predicted attention (i.e., a temporal trajectory, saliency maps, etc.). In the deep learning era, many neural models (Cornia et al., 2016; Palazzi et al., 2019) have surpassed classic methods (Itti et al., 1998). Recently, an unsupervised dynamical model (Zanca et al., 2020) has been proposed for attention trajectory. It can be applied both to static images and videos and has been studied in the context of online learning in deep networks (Tiezzi et al., 2020). Without any loss of generality, this is the model we consider in this chapter.

**Learning motion-invariant features.** According to Feldman and Tremoulet (2006), humans identify objects based on the consistency of a subset of their associated features during movement. Some brain-inspired neural networks disentangle *what* features (identity and semantics) and *where* features (motion and location), each of them separately encoding informative and uninformative factors of variation (Burt et al., 2021). Then, in the same sense, we can think of developing *what* visual features that are invariant with respect to the apparent movements of a given object. We develop a framework where attention trajectory (Sec. 6.4) and motion flow (Sec. 6.5) implicitly constrain the agent to learn such invariances, resulting in features that keep constant across time and space. This idea is linked to recent studies about learning invariance to motion in unsupervised learning over time (Betti et al., 2020b).

**Semantic segmentation.** Semantic segmentation in computer vision aims at associating each pixel of a given image with a class label. Deep architectures for this

task usually rely on supervised learning from offline data, including fully convolutional networks (Long et al., 2015), models based on transposed convolutions, dilated convolutions, upsampling and/or unpooling (U/V-net architectures (Ronneberger et al., 2015)), transformers (Ranftl et al., 2021). We will make comparisons with these models. However, for our task, we’ll use the term *pixel-wise classification* instead of *semantic segmentation*. This is because our final goal is not to describe every pixel in the agent’s visual universe, obtained through exhaustive labelling of the environment. Instead, we aim to precisely identify pixels that belong to specific object classes, which are progressively defined over time through supervision.

### 6.3 Learning and evaluation setting

Let us assume the availability of an endless collection of subsequent frames from a visual source,  $\{I_t | t > 0\}$ , originating a video stream  $\mathcal{V}$  at the resolution of  $w \times h$ , with  $Z^\diamond := \{1, \dots, w\} \times \{1, \dots, h\}$  the set of valid pixel coordinates for any frame. Each spatial location  $x \in Z^\diamond$  of  $I_t$  can be associated with a feature vector, carrying information about what is present in such a location and its neighborhood. We talk about *pixel-wise feature maps* when referring to the collection of such feature vectors. We focus on the challenge of progressively developing a robust feature extractor  $\mathcal{F}$  leveraging the information in  $\mathcal{V}$ , working in a continual online manner, without storing buffers of past data, and without any supervisions. The learnt features can then be used to preprocess visual stimuli in the context of different downstream tasks. It is important to remark that frames are continuously streamed at a constant frame rate, without any temporal limits ( $t$  could be potentially  $\infty$ ) and their content smoothly changes over time, thus they are *not* independent. Under these conditions, effective learning of useful features over time is not trivial.

#### 6.3.1 Feature evaluation: pixel-wise classification

In this chapter, we assume that there is a visual agent, consuming the video stream  $\mathcal{V}$ , and a human supervisor, occasionally interacting to provide supervision. In order to simplify interaction, both the supervisor and the agent have access to some attention trajectory, i.e., for each frame, the coordinates  $a_t$  of the attention marker. The human supervisor occasionally provides a supervision at coordinates  $a_t$ , asserting the membership of the corresponding pixel in a certain class  $y_t$ .<sup>3</sup>

Let us define an open-set classifier  $c(\cdot, \xi, \zeta)$  that predicts the class-membership scores from a pixel-wise feature vector, over a certain number of classes. The main parameters of the classifier are collected in  $\zeta$ , and when all the membership scores

<sup>3</sup>Notice that only one pixel gets a supervision at time  $t$ , thus it is different from few-shot semantic segmentation (Min et al., 2021).

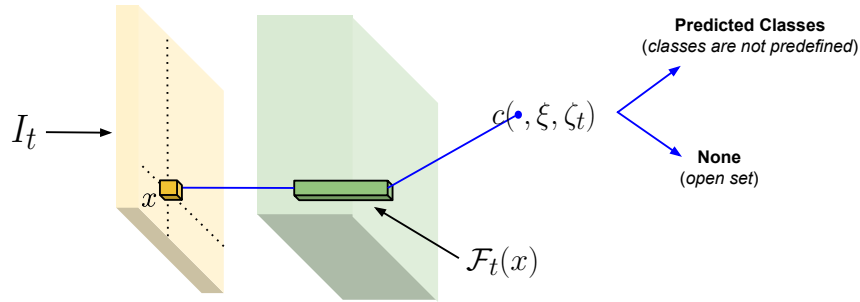


Figure 6.2: Pixel-wise classification. Pixels of frame  $I_t$  are encoded into new representations by the feature extractor  $\mathcal{F}$ . Inference happens independently on all the coordinates, and the classifier  $c$  can predict one or more classes or nothing (open-set). Classes are not known in advance. Development of features does not depend on supervisions.

are below threshold  $\zeta$  the classifier assumes to be in front on an unknown visual element and it does not provide a decision (*open-set*)—see Fig. 6.2.

Whenever a supervision on a never-seen-before class is received, the classifier becomes capable of computing the membership score of such class for all the following time steps (*class-incremental*). We consider the case in which supervisions are extremely rare, not offering a suitable basis for gradient-based learning of the classifier nor the feature extractor. Clearly, some other learning process has to occur, in order to learn compact representations that can be easily classified by the aforementioned classifier.

The most straightforward way to implement the open-set classifier  $c$  is with a distance-based model, storing the feature vectors associated to the supervised pixels as templates.<sup>4</sup> This allows the model to abstain from prediction when the minimum distance from all the templates is greater than a certain threshold (related to  $\zeta$ ). We indicate with  $(k_t, y_t)$  a supervised pair where  $k_t$  is the feature vector extracted at coordinates  $a_t$  from frame  $I_t$ . In principle, if the agent is allowed to continue its autonomous development, templates may become outdated due to the evolution of the system, leading to potentially wrong predictions. However, we are assuming that supervision is extremely scarce (i.e., a few pixels per entity). This means that the agent can easily store the frames containing the supervisions for periodical refresh of the templates, implementing in this case a simple instance of rehearsal strategies in continual learning (Sec. 2.2).

<sup>4</sup>We tested the squared Euclidean distance and cosine similarity.



### 6.3.2 Comparison

We compared the obtained features against those produced by massively pretrained state-of-the-art models. We considered the Dense Prediction Transformer (DPT) (Ranftl et al., 2021) and DEEPLABV3 (Chen et al., 2017) with ResNet101 backbone, testing both the features produced by the penultimate layer in the classification heads (-C suffix) and the ones obtained by the backbones (i.e., upsampling the representations if needed, -B suffix). In this way, we investigate both lower-level features based on backbones pretrained on millions of images (ImageNet), and task-specialized higher-level features for semantic segmentation (COCO (Lin et al., 2014) and ADE20k (Zhou et al., 2019) datasets—the latter explicitly includes the categories of the considered textured objects). We also considered the features from the pre-trained backbones of recent self-supervised approaches, including MoCo v1, v2, v3 (He et al., 2020; Chen et al., 2020b, 2021) and PixPro (Xie et al., 2021). We upsampled the features picked at one of the last three residual stages (testing all of them) and reported the best-performing configuration. More details are available in Appendix C.1. As BASELINE model we considered the case in which the pixel representations are left untouched (i.e., pixel color/brightness).

### 6.3.3 Video streams

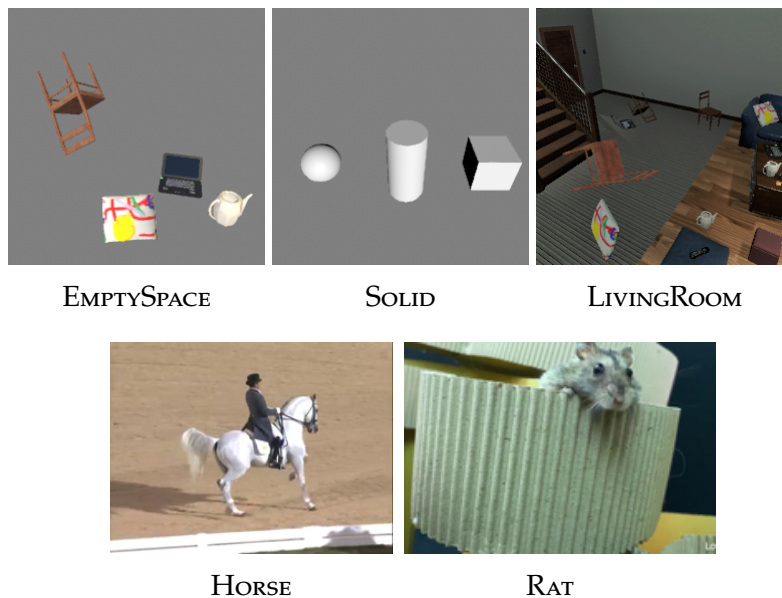


Figure 6.3: Sample frames from the three 3D scenes (objects rotate and scale while moving) and from real-world videos.

We used photo-realistic 3D Virtual Environments within the SAILenv platform (Meloni et al., 2021), that provides pixel-wise semantic labeling and motion flows

of potentially endless streams. We created three 3D scenes to emphasize different and challenging aspects on which we measure the skills of the agents.<sup>5</sup> The agent observes the scene from a fixed location, and some objects of interest move, one at the time, along pre-designed smooth trajectories while rotating and getting closer to/farther from the camera. We denote with the term *lap* a complete route traveled by each object to its starting location.

We designed three different scenes. (i) `EMPTYSPACE`: four photo-realistic textured objects from the `SAILenv` library (chair, laptop, pillow, ewer) move over a uniform background. The goal is to distinguish them in a non-ambiguous setting. (ii) `SOLID`: a gray-scale environment with three white solids (cube, cylinder, sphere) is considered. Due to the lack of color-related features, the agent must necessarily develop the capability of encoding information from larger contexts around each pixel. (iii) `LIVINGROOM`: the objects from `EMPTYSPACE` are placed in a photo-realistic living room composed by other non-target objects (i.e., an heterogeneous background with a couch, tables, staircase, door, floor), and multiple static instances of the objects of interest. Samples are shown in Fig. 6.3 (top). We created three pre-rendered 2D visual streams by observing the dynamic scenes ( $256 \times 256$  pixels, with  $\approx 51$  k, 12 k and 20 k frames, respectively, corresponding to 31 completed laps for each object), both in grayscale (BW) and color (RGB)—`SOLID` is BW only.

Notably, we also discuss experiments run with natural video streams. We consider two long real-world videos, `RAT` ( $256 \times 128$ ) and `HORSE` ( $256 \times 192$ ), proposed by Liang et al. (2020), that recorded the behaviour of a rat and a horse (actually horse+jockey), respectively. These videos<sup>6</sup> are used to assess the capability of the proposed methods to generalize to real-world scenarios, with rich visual textures, realistic illumination and non-fixed camera.

## 6.4 COAT: attention-based model

In this section we describe COAT (Consistency Over Attention Trajectory), a novel approach to online learning from a video stream. COAT is built around the idea of using a scanpath-based focus of attention mechanism (Zanca et al., 2020) to explore the video and to drive the learning dynamics in conjunction with motion information. Human-like attention trajectories has been recently proved to efficiently select the most salient information of the video stream when learning with deep architectures (Tiezzi et al., 2020). We propose to learn representations that remain consistent over the slow movements of the simulated attention gaze (Sec. 6.4.2), since those movements are likely to cover visual patterns with the same semantics (Rucci and Poletti, 2015; Rucci et al., 2016). Attention is paired with information coming

<sup>5</sup>Code, data and selected hyper-parameters can be downloaded as described in Appendix B, C.

<sup>6</sup><https://www.kaggle.com/datasets/gvclsu/long-videos>

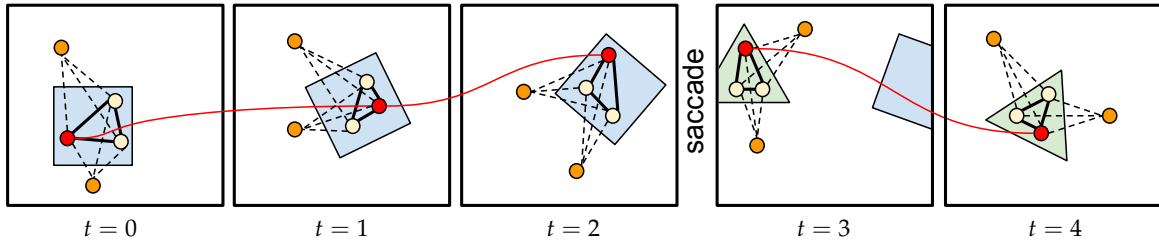


Figure 6.4: A rectangle rotates toward the right, attracting the attention  $a_t$  (red nodes) that explores it. Then, a triangle enters the scene, and the simulated gaze quickly moves on it (saccade). The red path links nodes on which we enforce *temporal consistency* (not across saccades). For each  $t$ , some coordinates (yellow) are sampled in the *moving* region that includes  $a_t$ , while other points (orange) are sampled outside of it. Solid lines (positive edges) link *spatially coherent* samples; dashed lines (negative edges) are about the *contrastive* term.

from motion (Sec. 6.4.3), that intrinsically suggests the spatial bounds (Sec. 6.4.4) of the attended area. This leads to a spatio-temporal unsupervised criterion that promotes coherence in the representations learned while observing what is moving (Fig. 6.4). In order to avoid trivial solutions, we augment the criterion with a contrastive term (Sec. 6.4.5) that favours the development of distinct representations of pixels that are inside the moving area compared to those of pixels that are right outside of it. Thanks to a graph-based formalization of this approach, we define a stochastic procedure (Sec. 6.4.6) that introduces variability in the information provided to the learning algorithm and leads to faster processing, also mitigating the effects of noisy motion information.

### 6.4.1 The model

Being  $X$  the set  $\mathbb{R}^{w \times h \times c}$ , the feature extractor  $\mathcal{F}$  is simply a neural network model that implements the function  $f(\cdot, \theta): X \rightarrow F$ , with  $F = \mathbb{R}^{w \times h \times d}$ , where  $\theta$  indicates the weights and biases of the network. As such,  $f(I_t, \theta)$  is an encoded representation of the frame where, for each pixel, we have a vector with  $d$  components (Fig. 6.2). The network evolves over time and  $\theta_t$  are the weights and biases at time  $t$ . In the following, we will use the shortcut  $f_t := f(I_t, \theta_t)$  to denote the features extracted at time  $t$ . For the sake of simplicity,  $f_t(x)$  is the feature vector picked for the pixel at 2D coordinates  $x \in Z^\diamond$ . In online learning from a video stream  $\mathcal{V}$ , the network weights  $\theta_{t+1}$  are obtained updating the previous  $\theta_t$ , using a law that depends on the gradient of a suitable loss function  $L$  with respect to parameters  $\theta$ . Such loss function  $L$  implements *temporal* and *spatial consistency* constraints to encourage pixel representations to be consistent in time and space, with respect to locations virtually connected by the attention trajectory and by motion, respectively.

### 6.4.2 Human-like attention

The first pillar of our model is a function  $t \mapsto a(t)$ , yielding an explicit estimate of the 2D coordinates where humans would focus the attention<sup>7</sup> at each time  $t$ . Among the various attention models yielding temporal trajectories (Borji and Itti, 2012), the recent unsupervised model by Zanca et al. (2020) achieved state-of-the-art results in simulating human-like attention. The authors showed that visual cues, such as texture and motion, can act as gravitational masses. The equation of the potential of the gravitational field,  $\varphi(x, t) := -(2\pi)^{-1} \int_Z \log \|x - z\| \mu(z, t) dz$ , is at the basis of the attention model, where  $Z$  is the continuous set of frame coordinates and  $\mu(x, t)$  is the total mass at  $(x, t)$ . In particular, the magnitude of the brightness and an optical-flow-based measure of motion activity are modeled as masses, and their impact is controlled by two positive scalars  $\alpha_b$  and  $\alpha_m$ , respectively. The focus of attention is modeled as a point-like particle subject to the above potential (including inhibition of return), and its trajectory  $a(t)$  is determined integrating the following equation with initial conditions  $a(0) = a^0$  and  $\dot{a}(0) = a^1$ ,

$$\ddot{a}(t) + \mu \dot{a}(t) + \nabla \varphi(a(t), t) = 0 \quad t > 0, \quad (6.1)$$

where the dissipation is controlled by  $\mu > 0$  and  $\nabla \varphi$  is the spatial gradient of the potential. Patterns in human-like attention have been extensively studied (Kowler, 2009): we introduce here a few notions that will be exploited in the following. It is known that the gaze performs *fixations* in locations of interest, with relatively low speed movements. *Smooth pursuit* consists of slow tracking movements performed to follow a considered stimulus. Differently, *saccades* are fast movements to relocate the attention toward a new target. The first two categories of patterns can be approximately separated from the latter exploiting a thresholding procedure on gaze velocity, based on some  $v > 0$ .

### 6.4.3 Temporal consistency

During fixations, the attention spans a certain part of an object, with limited displacements of the gaze. Similarly, during smooth-pursuit the attended moving area has uniform semantic properties. Differently, during saccades, the attention switches the local context, shifting toward something that might or might not belong to the same object (Rucci et al., 2016). Assuming the validity of these notions in case of artificially generated scanpaths, we implement *temporal consistency* by defining the

---

<sup>7</sup>In this chapter, we will only consider free-viewing exploration, i.e., the viewer is not driven by any specific task but simply exposed to a novel environment. In such conditions, experimental evidence shows that the attention trajectory adheres to broadly consistent and predictable patterns (Zanca and Gori, 2017).

loss function<sup>8</sup>  $L_T$ ; such loss function (*i.*) restricts learning to the attention trajectory, filtering out the information in the visual scene (Tiezzi et al., 2020) and (*ii.*) avoids abrupt changes in the feature representation during fixations and smooth pursuit,

$$L_T(f_t, f_{t-1}) := \pi_t \|f_t(a_t) - f_{t-1}(a_{t-1})\|^2, \quad (6.2)$$

where  $\|\cdot\|$  is the Euclidean norm<sup>9</sup>, and  $\pi_t$  is equal to 0 in case of saccades, otherwise it is 1. This loss will be paired with other penalties described in the following, thus avoiding trivial solutions.

#### 6.4.4 Spatial consistency

Temporal consistency is not enough to capture the spatial extension of visual stimuli, since it is only limited to the attention trajectory. Motion naturally provides such spatial information, and we follow this intuition designing agents that learn by observing moving attended regions.<sup>10</sup> As a matter of fact, motion plays a twofold role, being crucial in defining the attention masses and as a mean to extend the notion of consistency to a frame region, i.e., *spatial consistency*. Formally, for each fixed  $t \in \mathbb{N}$ , we indicate with  $S_t \subseteq Z^\diamond$  the set of frame coordinates that belong to the region of connected moving pixels that includes  $a_t$ .<sup>11</sup> We introduce what we refer to as *spatial attention graph* at  $t$ ,  $\mathcal{G}_t$ , with a node for each pixel of the frame ( $\forall x \in Z^\diamond$ ) and with two types of edges, referred to as positive and negative edges. Positive edges link pairs of nodes whose coordinates belong to  $S_t$ , while negative edges link nodes of  $S_t$  with nodes outside the moving region. The positive edges of the attention graph allow us to introduce a spatial consistency loss  $L_S$ ,

$$L_S(f_t) := \frac{1}{2} \sum_{x, z \in S_t, x \neq z} \|f_t(x) - f_t(z)\|^2, \quad (6.3)$$

that encourages the agent to develop similar representations inside the attended moving area  $S_t$ . The notion of learning driven by the fulfilment of spatio-temporal consistency over the attention trajectory ( $L_S$  and  $L_T$  of Eq. 6.2 and Eq. 6.3) is the second pillar on which our model is built.

#### 6.4.5 Contrastive loss

In order to prevent the development of trivial uniform solutions, which fulfill the spatio-temporal consistency, we add a *contrastive* loss  $L_C$  that works the opposite

<sup>8</sup>For the sake of readability, we will always omit the dependency of the loss function on parameters  $\theta$  of the neural model.

<sup>9</sup>We also consider the case of feature vectors with unitary Euclidean norm—see Appendix B.

<sup>10</sup>In the case of egomotion (i.e., camera moves over time), filtering out camera motion with hardware or software techniques is appropriate.

<sup>11</sup>Moving pixels are detected as detailed in Appendix B.

way  $L_S$  does. In particular,  $L_C$  exploits negative edges of  $\mathcal{G}$  to foster distinct representations of pixels inside the moving area compared to the ones of pixels outside of it,

$$L_C(f_t) := \left( \sum_{x \in S_t, z \in O_t} \|f_t(x) - f_t(z)\|^2 + \varepsilon \right)^{-1}, \quad (6.4)$$

where  $O_t = Z^\diamond \setminus S_t$  is composed of frame coordinates not in  $S_t$  and  $\varepsilon > 0$  avoids divisions by zero. We notice that this contrastive loss has a different structure from InfoNCE losses (see Sec. 2.3 and Sec. 6.5).

### 6.4.6 Stochastic spatio-temporal consistency

These spatial and contrastive losses are plagued by two major issues. First, the number of pairs in the summations of Eq. 6.3 and Eq. 6.4 is large, being it  $(1/2)|S_t|(|S_t| - 1)$  and  $|S_t|(wh - |S_t|)$ , respectively, making the computation of the loss terms pretty cumbersome. Secondly, whenever an exact copy of a moving object appears in a static part of the scene, there will be pixels of the first instance that are fostered to develop different representations with respect to pixels of the second one, what we refer to as *collision*. However, this clashes with the idea of developing a common representation of the object pixels. Hence, we replace  $\mathcal{G}_t$  with the subgraph  $\tilde{\mathcal{G}}_t$ , that is the *stochastic spatial attention graph* at time  $t$ , composed by nodes that are the outcome of a stochastic sub-sampling of those belonging to  $\mathcal{G}_t$ . In particular, the node set of  $\tilde{\mathcal{G}}_t$  is the union of  $\tilde{S}_t \subseteq S_t$  and  $\tilde{O}_t \subseteq O_t$ , where  $a_t$  is guaranteed to belong to the former. Edges of  $\tilde{\mathcal{G}}_t$  are the positive and negative edges of  $\mathcal{G}_t$  connecting the subsampled nodes. The key property of the stochastic graph is that the number of positive and negative edges is a chosen  $e > 0$ . The set  $\tilde{S}_t$  is populated by uniformly sampling nodes in  $S_t$ , ensuring that  $a_t$  is always present, while  $\tilde{O}_t$  is populated by sampling from a Gaussian distribution centered in  $a_t$  and with variance  $\sigma$ , discarding samples  $\notin O_t$ .<sup>12</sup> Large  $\sigma$ 's lead to sampling data also far away from the focus of attention, while small  $\sigma$ 's will generate samples close to the boundary of the moving region. In the example of Fig. 6.4 we emphasized how the attention bridges multiple instances of  $\tilde{\mathcal{G}}_t$  over time, yielding a *stochastic attention graph*. Such a graph reduces the probability of collisions, both due to the random sampling and to the control on the sampled area by means of  $\sigma$ , and it introduces variability in the loss functions also when computed on consecutive frames. Moreover, the impact of imperfect segmentation of the moving region is reduced, since only some pixels are actually exploited, re-sampled at every frame. Since the number of pairs is bounded, the stochastic graph makes the formulation suitable for real-time processing. Vary-

<sup>12</sup>We selected  $\sigma$  to be  $\propto \sqrt{|S_t|}$  by means of an integer spread factor  $\beta \geq 1$ . We repeat the Gaussian sampling until we collect the target number of points in  $\tilde{O}_t$ , up to a max number of iterations.

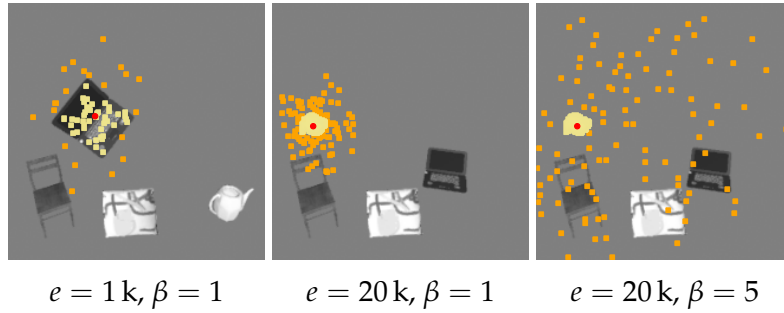


Figure 6.5: Samples of stochastic spatial graphs in `EMPTYSPACE`, varying related hyper-parameters (same color patterns of Fig. 6.4).

ing the above introduced parameters  $e$  and  $\beta$ , we can model stochastic consistency graphs with different properties, as showed in Fig. 6.5.

### 6.4.7 Cumulative loss

We define  $L$  as the cumulative loss at a certain time instant  $t$ , where the contributions of Eq. 6.2, Eq. 6.3, Eq. 6.4 are weighted by the positive scalars  $\lambda_T$ ,  $\lambda_S$  and  $\lambda_C$ ,

$$L(f_t, f_{t-1}) := \lambda_T L_T(f_t, f_{t-1}) + \lambda_S \tilde{L}_S(f_t) + \lambda_C \tilde{L}_C(f_t). \quad (6.5)$$

The losses  $\tilde{L}_S$  and  $\tilde{L}_C$  are the stochastic counterparts of  $L_S$  and  $L_C$ , respectively, in which the sets  $\tilde{S}_t$  and  $\tilde{O}_t$  are used in place of  $S_t$  and  $O_t$ . We define  $\nabla L$  to be the gradient of the loss with respect to neural network parameters  $\theta_t$ , that drives the online learning process,  $\theta_{t+1} = \theta_t - \alpha \nabla L(f_t, f_{t-1})$ , with  $\alpha > 0$  (learning rate) and  $\theta_0$  some random initialization of the parameters of the network.

## 6.5 CMOSFET: motion-based model

Motivated by the considerations about the paramount role of motion in learning visual skills mentioned in Sec. 6.1, we propose to move a step forward. Here we face the feature learning problem with a neural architecture designed to jointly learn to estimate motion and extract motion-conjugated features (Betti et al., 2022b). This idea is also inspired by recent studies (Milner, 2017) about brain functioning, suggesting that neural pathways for identifying objects and understanding their spatial interactions might be much more intertwined (continuous cross-talk) than what previously thought (Cloutman, 2013). In our method, the learning process happens in a continual manner, so that we refer to our method as Continual MOtion-based Self-supervised Feature EXtractor (CMOSFET).

Noteworthy, CMOSFET also learns motion estimation in a continual unsupervised setting, inheriting the experience described in Ch. 5. COAT model, proposed in Sec. 6.4, is very related to the idea that we are going to describe, since it learns

	COAT	CMOSFET
<i>Self-Supervision signal</i>	Attention and motion	Motion
<i>External components</i>	Attention	None
<i>Learning principle</i>	Spatio-temporal consistency	Motion-feature conjugation
<i>Contrastive mechanism</i>	Inside vs outside of attended moving blob	Things that move in different directions
<i>Contrastive loss</i>	Custom	InfoNCE-like
<i>Computational structure</i>	Plain	Multi-level
<i>Learning dynamics</i>	Online	Online with fast/slow learners
<i>Sampling</i>	Centered on focus of attention	Based on motion and features

Table 6.1: Informal summary of the differences between the two proposed methods, COAT and CMOSFET.

online from a video stream. However, CMOSFET has a more complex structure and does not rely on external components (the attention trajectory). In order to easily grasp the difference between COAT and CMOSFET, the reader can refer to Tab. 6.1.

Following the deep and compositional nature of modern neural architectures for computer vision, we assume that processing the stream  $\mathcal{V}$  with model  $\mathcal{F}$ <sup>13</sup> results in extracting pixel-wise features at multiple ( $N \geq 1$ ) levels of abstraction, indexed by  $\ell \in [1, N]$ . In CMOSFET, a key role is played by the way in which we use optical flow. In Ch. 5 we introduced optical flow and showed that networks for flow extraction can be trained in a continual online fashion, without replay buffers or specific continual learning methodologies. However, in the proposed framework, not only optical flow is estimated in a continual learning fashion, as in Ch. 5, but also higher-level flows (generalization of optical flow, tracking higher-level features) are continuously estimated.

Our approach processes frames in an online manner with a two-branch neural architecture, continuously-and-jointly learning to extract pixel-wise motion fields and pixel-wise visual features, in a self-supervised manner. Our contrastive criterion (Sec. 6.5.3) revisits the definition of positive and negative pairs, establishing spatio-temporal correspondences/contrasts among pixels belonging to consecutive frames. Similarly to what we have done in Sec. 6.4, we make learning affordable by restricting the contrastive loss to a group of stochastically selected pixels.

Motion consistency (Sec. 6.5.2) naturally regularizes feature learning over time, and we gain further stability and reduce catastrophic forgetting by introducing a fast learner / slow learner implementation (Sec. 6.5.4), where the fast learner is driven

<sup>13</sup>Due to the more articulated structure of the approach described in this section, we use the notation  $f$  for the elementary feature extractor and  $\mathcal{F}$  for the overall feature-extraction model.



by gradient steps while the slow learner model smoothly adapts to the video stream via a momentum-based update scheme.

Notably, motion is the only driving signal for the development of the visual feature extractor, which conquers invariance properties naturally induced by the motion fields.

### 6.5.1 Multi-level feature flows

We indicate with  $f^\ell$  the feature extractor of the  $\ell$ -th level, consuming the output of  $f^{\ell-1}$  as input, see Fig. 6.6 (green boxes). The notation  $f_t^\ell(x)$  indicates the feature

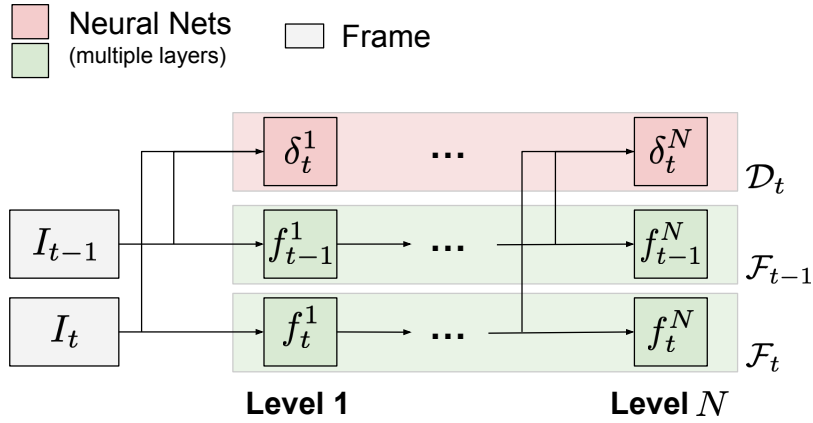


Figure 6.6: The architecture of CMOSFET. Given a pair of consecutive frames  $I_{t-1}$  and  $I_t$ , pixel-wise features  $(f_{t-1}^\ell, f_t^\ell)$  and motion flow  $\delta_t^\ell$  are extracted at multiple levels, indexed by  $\ell$ .

vector of length  $d^\ell$  from level  $\ell$  at the 2D spatial coordinates  $x$  and timestep  $t$ , while  $\mathcal{F}_t = \{f_t^\ell, \ell = 1, \dots, L\}$  is the collection of all the output features, with  $\ell = 0$  being the degenerate case ( $f_t^0 = I_t$ ). We now introduce the generic notion of *feature flow* as a generalization of optical flow when associated to arbitrary visual features  $f_t^\ell$ , including  $I_t$  and more abstract features, motivated by the theoretical insights of Betti et al. (2022b). The notation  $\delta^\ell$  is used to indicate the function that extracts the  $\ell$ -th-level feature flow, being  $\delta_t^\ell(x)$  the flow vector at location  $x$  and timestep  $t$ . Here and in the rest of the section, for the purpose of conciseness, the aforementioned symbols  $f$  and  $\delta$  will be overloaded depending on the specific context in which they are exploited, using them to denote functions or to represent their respective outputs, without any ambiguities.

It is convenient to represent motion in terms of displacement vectors between two consecutive frames, so that the feature flow  $\delta_t$  establishes a point-wise correspondence between some features at  $f_{t-1}$  and  $f_t$ , and it allows the approximate re-

construction of  $f_{t-1}$  given  $f_t$  (or vice-versa).<sup>14</sup> Formally, neglecting the superscript  $\ell$  and introducing the *warping* operator  $W$ :

$$f_{t-1}(x) \approx W(f_t, \delta_t)(x) := f_t(x + \delta_t(x)) \quad (6.6)$$

The flow  $\delta_t$  can be computed by a neural network  $\delta$  that processes the representations of a pair of frames, at time  $t - 1$  and  $t$ , following Ch. 5. We cast this principle into the  $N$ -levels architecture described so far, so that  $\delta_t^\ell$  is the feature flow at level  $\ell$ , and it depends (see Fig. 6.6, red boxes) on the frame representations produced at the level below<sup>15</sup>,  $f^{\ell-1}$ . Along with the feature extractors in  $\mathcal{F}$ , we use the notation  $\mathcal{D}$  to indicate the collection of all the neural networks  $\delta^{\ell'}$ s, dedicated to the computation of flows.

The classic optical flow has a clear meaning related to apparent motion of objects with respect to the camera. However, it is largely known that this correspondence loses its strict physical interpretation when considering non-ideal conditions (e.g., strong occlusions, non-textured objects, changes of lighting, etc.) (Horn and Schunck, 1981). This consideration becomes very precious when interpreting the feature flows  $\delta_t^\ell$  with  $\ell \geq 1$ . In deep architectures, higher-level features typically correspond to increasingly abstract representations that emerge from the learning process. As a result, it is challenging to explicitly define expectations for  $\delta_t^\ell$ . Additionally, higher-level features tend to encode information from a larger neighborhood around a given location in  $I_t$  (see receptive fields in convolutional networks), causing local changes in  $I_t$  to potentially affect feature vectors associated with distant locations. Consequently, our multi-level flows are not accompanied by predetermined interpretation but rather serve as latent signals that facilitate the transfer of information among motion-connected locations, as we will delve into shortly.

## 6.5.2 Flow-conjugated representations

Several studies have investigated the idea of acquiring features that align with motion (Bregler, 1997; Xiong et al., 2021; Pathak et al., 2017). Recently, Betti et al. (2022b) introduced the notion of feature fields which are “conjugate fields with respect to motion”. This concept is characterized by constraints that specify the connections between a collection of arbitrary pixel-wise characteristics and the motion signal. Following such a work, we consider the bidirectional *conjugation* between features and their respective flows of Eq. 6.6 as an essential desirable property for learning purposes. On one hand, fulfillment of the approximation in Eq. 6.6 can be interpreted as the constraint that enforces flows to be consistent with respect to

<sup>14</sup>We removed the layer index on purpose, since this description is generic. As it will become clear shortly, multiple feature levels can be associated to the same flow.

<sup>15</sup>Notice that we defined  $f^0(x) := I(x)$  for any timestep, i.e., the original pixel representations.

some “given features”, as in classic optical flow, where the features are brightness levels or colors. On the other hand, it stimulates “learnable features” on consecutive frames to be consistent with some estimated flow, that is the dual of plain optical flow, where features are given (visual input  $I_t$ ). Formally, we define the following consistency penalty  $\mathcal{L}_c$  for the constraint of Eq. 6.6,

$$\mathcal{L}_c(\delta_t, f_{t-1}, f_t) = \frac{1}{wh} \sum_x \rho(f_{t-1}(x) - W(f_t, \delta_t)(x)), \quad (6.7)$$

being  $\rho$  a differentiable penalty function, that we selected to be the generalized Charbonnier photometric distance,  $\rho(a) = (\|a\|^2 + \epsilon)^\zeta$  (Sun et al., 2010), with  $\epsilon = 0.001$  and  $\zeta = 0.5$ , as in Ch. 5.

**Conjugation Loss** The consistency penalty of Eq. 6.7 represents a generic loss that could be straightforwardly exploited for learning purposes, using the features and flows yielded at each level. However, such implementation would not introduce any explicit cross-layer relation between the two branches (apart from the one arising from the cascaded feature computation). We propose to exploit three instances of the penalty in Eq. 6.7 to setup a loss function<sup>16</sup> that shapes the way features and flows are developed across the model,

$$\begin{aligned} L_{conj}^\ell = \mathcal{R}(\delta_t^\ell) & \\ & + \mathcal{L}_c(\delta_t^\ell, f_{t-1}^\ell, f_t^\ell) \quad \bullet \text{ (i.)} \\ & + \mathcal{L}_c(\delta_t^1, f_{t-1}^\ell, f_t^\ell) \quad \bullet \text{ (ii.)} \\ & + \mathcal{L}_c(\delta_t^\ell, f_{t-1}^{\ell-1}, f_t^{\ell-1}), \quad \bullet \text{ (iii.)} \end{aligned} \quad (6.8)$$

where  $\mathcal{R}(\delta_t^\ell)$  is the usual regularization term of optical flow, proportional to the sum of squares of the spatial gradient of the flow,  $\nabla \delta_t^\ell$ , and required for the well-position of the flow extraction problem (Horn and Schunck, 1981). The role of the other three operands in the sum of Eq. 6.8 is illustrated in Fig. 6.7, and described in the following: (i.) is the consistency between features and flow computed at layer  $\ell$ , as expected; (ii.) is the consistency of the features of layer  $\ell$  and motion from the first level, encouraging  $f^\ell$  to be compatible with the displacement yielded by classic optical flow—i.e., learning promotes the preservation of a connection between higher-level features and the lowest-level motion, which is more directly associated with the fine details of the original image; (iii.) is the consistency between flow estimated in level  $\ell$  with the features of the previous level, to encourage higher-level motion flows to be compatible with less abstract features. We notice that the term *iii.* is the usual loss for optical flow estimation, applied at the different levels of the

<sup>16</sup>For the sake of readability, we omit the scaling factors in front of each term in the summation. Also, we will always omit the dependency of the loss functions on parameters of the neural model.

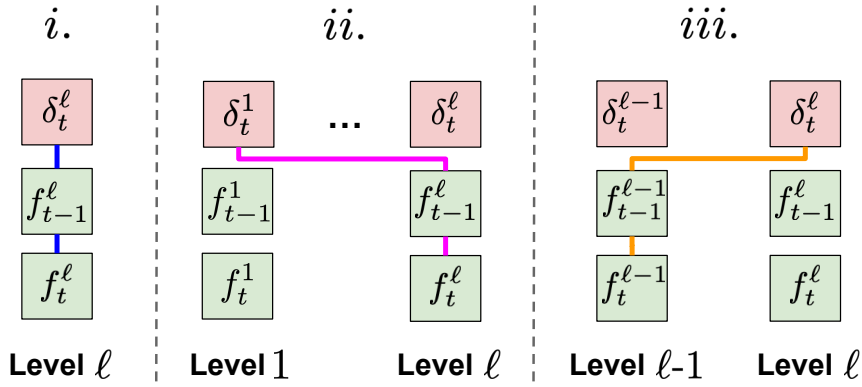


Figure 6.7: Illustration of the three  $\mathcal{L}_c$  terms (*i.*, *ii.*, *iii.*) in Eq. 6.8. Each of the three sub-pictures include portions of the architecture of Fig. 6.6, while connections indicate the dependency introduced by the  $\mathcal{L}_c$  term. Case *i.* is about a single level, while *ii.* and *iii.* introduce cross-level dependencies.

feature hierarchy. It is interesting to analyze the meaning of the conjugation loss in the case of  $\ell = 1$ , the lowest-level flow. Intuitively,  $\delta_t^1$  flow is closely related to classic optical flow, since keeping only *iii.* in Eq. 6.8 makes it equivalent to the usual loss for optical flow estimation (recall that  $f_t^0$  is  $I$ ). Moreover, terms *i.* and *ii.* degenerate to the same term. To enforce this analogy and supported by preliminary experiments, we stop gradients to propagate to  $\delta$  through the *i.* and *ii.* losses, that allows our model to guarantee that  $\delta_t^1$  is coherent with the classic optical flow, that also properly drives the development of  $f_{t-1}^\ell$  and  $f_t^\ell$  for all the levels of the network.

### 6.5.3 Self-supervised contrastive learning

Despite these feature-flow constraints, the objective of Eq. 6.8 can be easily minimized by trivial solutions (e.g., spatially uniform features with arbitrary flow), similarly to what happens in COAT. Then, we propose to introduce a flow-driven contrastive loss that aligns with the core principles of this section and encourages the emergence of meaningful features. Unlike popular contrastive approaches that are aimed at image-level global features (see Sec. 2.3), here we focus on the case of pixel-wise representations, thus we reconsider the role of positive and negative examples. Our idea consists in following the well-known Gestalt principle which sensibly states that things that move together often carries similar semantic information. Furthermore, our objective is to extend this concept to the scenario of learning from a visual stream, where the goal is to establish relationships not only between representations extracted at a single time instant but also across consecutive frames. In order to formally present our contrastive loss, we need to introduce three further ingredients, that are a similarity function  $s$ , a way to identify positive and negative pairs, a sampling strategy to make learning affordable. We will avoid specifying the

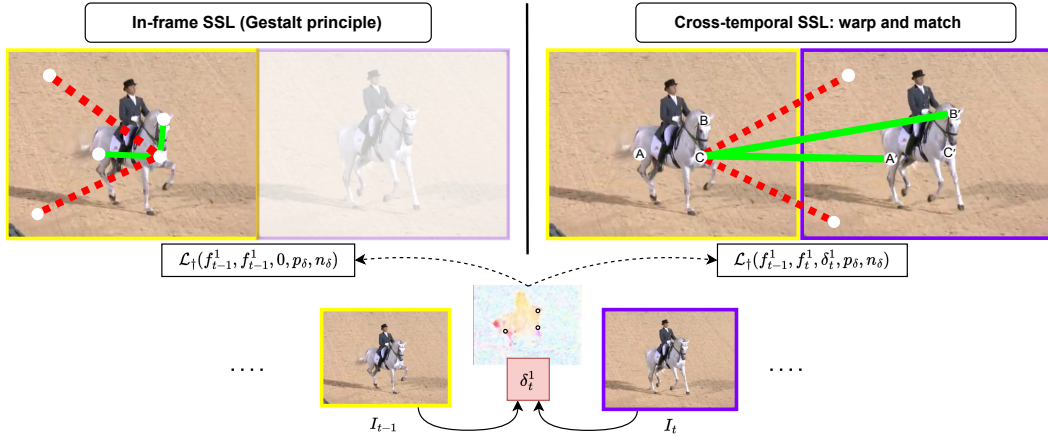


Figure 6.8: Self-supervised loss, considering a pair of consecutive frames ( $\bullet I_{t-1}$ ,  $\bullet I_t$ ), at the first level of the feature hierarchy ( $\ell = 1$ , notice that it holds for any  $\ell$ ). The objective  $L_{self}^1$  (Eq. 6.11) is the sum of two contrastive penalties computed through  $L_{\dagger}$ . The first contribution (left) encourages the development of features by comparing features extracted on the same frame ( $I_{t-1}$ ), while the second term (right) encourages alignment between features extracted in a pair of frames ( $I_{t-1}$ ,  $I_t$ ), thanks to flow matching (warping coordinates of items on one side of (dis)similarity relationships, e.g.  $A \rightarrow A'$ ,  $B \rightarrow B'$ ,  $C \rightarrow C'$ ). Green (red) links connect pixels whose features are encouraged to be similar (different) according to our motion-based criterion.

layer index  $\ell$  unless needed.

### Similarity Function

The similarity function  $s$  is responsible of comparing features for a generic pair of pixel coordinates  $(x_i, x_k)$ . Such features are obtained by extractors that we will generically indicate with  $g$  and  $h$  for the first and second element of the pair, respectively. The reason we introduce two different extractors will be made clear in the following and is related to the learning dynamics. In fact, we will consider the case in which  $x_i$  and  $x_k$  belong to the same frame, hence  $g \equiv h$ , and the case in which they belong to consecutive frames at  $t - 1$  and  $t$  ( $g \neq h$ ). In the former case, no warping of the pixel coordinates will be involved, that is equivalent to consider a degenerate instance of the warping function with null displacement for all its inputs. In the latter case,  $x_k$  is warped by motion  $\delta(x_k)$  before the feature extraction, i.e., we retrieve the location of the warped pixel  $x_k$  according to the estimated flow  $\delta$ , before extracting features. Formally,

$$s(x_i, x_k, \delta, g, h) := \tau^{-1} \text{sim}(g(x_i), h(W(x_k, \delta(x_k))))$$

that is the  $\tau$ -scaled cosine similarity with  $\text{sim}(a, b) := a \cdot b / (\|a\| \|b\|)$  and  $\|\cdot\|$  the Euclidean norm.

### Soft assignment of positive and negative pairs

Gestalt assumption (Wertheimer, 1938) is exploited to identify positive and negative pairs, considering as positive those pairs of pixels that move coherently, and as negative those pairs that move in a different manner. Of course, this is not expected to be strictly holding in practical scenarios, so we adopt a soft-assignment strategy to mark pairs depending on both distance and flow. Given a pixel at  $x$ , all the spatial coordinates of a given frame are evaluated in order to identify positive and negative examples. As a result, a pair that is composed of  $x$  and one of the *nearby* pixels with *similar motion* is a *positive pair*, while a *negative pair* is composed of  $x$  and one of the *distant* pixels with *different motion*. While this is a sensible learning signal, we need to mitigate the effect of the natural violations of the assumption (for example, a non-rigid object will have different motion patterns in different parts of its surface, there could be static clones of a moving instance, etc.). We indicate with  $p_\delta(x_i, x_j)$  the confidence score of  $(x_i, x_j)$  being a positive pair, while  $n_\delta(x_i, x_z)$  is the confidence score of  $(x_i, x_z)$  being a negative pair, each of them in  $[0, 1]$ . Scores  $n_\delta$  and  $p_\delta$  are computed as

$$\begin{aligned} n_\delta(x_i, x_k) &= [\text{sim}(\delta(x_i), \delta(x_k)) \leq \tau_n] \frac{\|x_i - x_k\|}{\sqrt{h^2 + w^2}}, \\ p_\delta(x_i, x_k) &= [\text{sim}(\delta(x_i), \delta(x_k)) > \tau_p] \left( 1 - \frac{\|x_i - x_k\|}{\sqrt{h^2 + w^2}} \right) \end{aligned} \quad (6.9)$$

where  $[\cdot]$  is 1 if the condition in brackets is true, otherwise it is 0 (Iverson bracket). Thresholds  $\tau_p, \tau_n \in [-1, 1]$  are selected to filter out pairs with uncertain similarities, with the condition  $\tau_p \geq \tau_n$  which ensures that  $p$  and  $n$  are non-zero in a mutually-exclusive manner. The rightmost operands of the products in Eq. 6.9 are the distance between  $x_i$  and  $x_k$  scaled in  $[0, 1]$  and the complement of such a distance, respectively. Noticeably, motion direction is not significant for static points (motion vector is smaller than a fixed threshold  $\tau_m$ ): a moving point and a static point are marked as dissimilar with maximum confidence score, independently on their distance, while a pair of static points is neither similar nor dissimilar. See also the illustration in Fig. 6.8, where positive pairs are connected by green lines, while red lines link negative pairs.

### Sampling

The number of positive and negative pairs might easily become large, since it is quadratic in the number of pixels, making the evaluation of the contrastive criterion computationally costly. For this reason, we introduce a motion-driven sampling procedure to make learning affordable, loosely inspired by Sec. 6.4. Here, we propose a selection criterion based on motion and representations, following the spirit of the section. The output of this procedure is a set of  $\eta$  coordinates  $\tilde{\mathcal{X}}_t \subset Z^\diamond$ , for each  $t$ ,

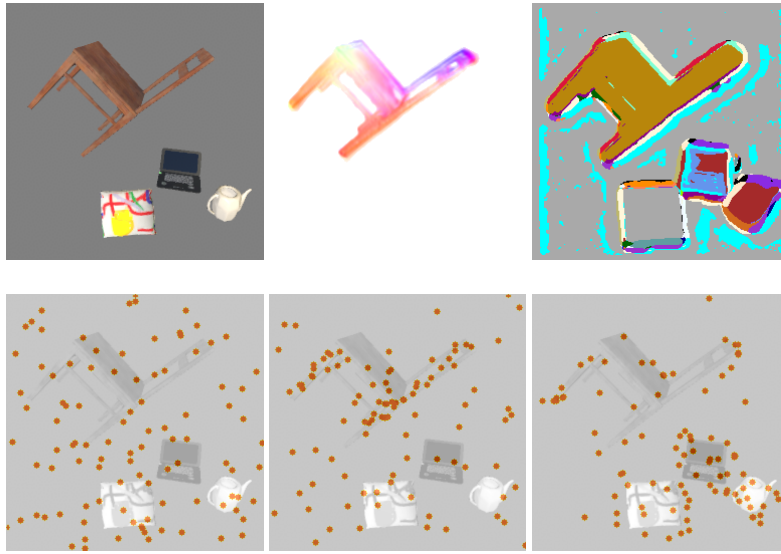


Figure 6.9: Illustration of different sampling strategies ( $\ell = 1$ ). Orange dots (bottom row) are points sampled from a visual stream (top row) in which a chair is moving in a static background that includes three smaller objects (pillow, laptop, teapot). First row: frame, estimated flow (different colors are about different directions), winning feature component (different colors are about different winning component). Second row, left-to-right: plain uniform sampling, motion-biased sampling, and the proposed sampling driven by both motion and winning components. In the last case, the sampled coordinates cover both the moving chair and other details of the image in a balanced manner, while the first and second case give more emphasis to the uniform region (first, second) or the moving object (second).

on which the contrastive loss will be evaluated. In detail, with the purpose of injecting priors into the sampling procedure, for any given pair of frames of  $\mathcal{V}$  from which we compute features and flows, we model an ad-hoc distribution over the pixel coordinates. Each pixel  $x$  at time  $t$  has probability  $P_t(x)$  of being sampled, with  $\sum_{x \in Z^\diamond} P_t(x) = 1$ . We define  $\{P_t(x), x \in Z^\diamond\}$  ensuring that (i.) the probability of sampling a point in moving areas is the same as sampling in static areas, and that (ii.) the probability of sampling in areas where the  $j$ -th feature component has the strongest activation (absolute value) is the same for all  $j$ 's. The rationale behind this idea is that (i.) we want to ensure that in shots with limited motion the sampling process is still biased toward moving areas (the informative ones), and (ii.) we want to ensure that different visual patterns are represented in a balanced way by the sampling procedure.

Formally,  $\mathcal{M}$  is the set of coordinates of moving pixels and  $\bar{\mathcal{M}}$  are the coordinates of the static pixels. We compute  $\mathcal{S}_j, j = 1, \dots, d$ , where set  $\mathcal{S}_j$  collects the coordinates of the pixels for which the  $j$ -th component of the learned representation is the largest

one, i.e.,  $\mathcal{S}_j = \{x_z: \arg \max |f_{t-1}(x_z)| = j\}$ <sup>17</sup>, where  $\arg \max a$  is intended to return the index of vector  $a$  associated to the largest component. It is trivial to see that each pixel coordinate  $x$  ends up in belonging to one of the following  $2d$  possible sets,

$$\{\mathcal{A}_j = \mathcal{S}_j \cap \mathcal{M}, \mathcal{A}_{2j} = \mathcal{S}_j \cap \bar{\mathcal{M}}, j = 1, \dots, d\}.$$

If we let  $P_t(x)$  be  $1/(2d \cdot \text{card}(\mathcal{A}_z))$ , being  $\mathcal{A}_z$  the set to which  $x$  belongs and  $\text{card}(\mathcal{A}_z)$  its cardinality,<sup>18</sup> we get a balanced distribution of the probabilities across the sets,  $\sum_{x \in \mathcal{A}_1} P_t(x) = \dots = \sum_{x \in \mathcal{A}_{2d}} P_t(x) = \frac{1}{2d}$ . Given  $P_t(x)$ , we sample  $\eta > 1$  pixel coordinates for each  $t$ , collecting their coordinates into  $\tilde{\mathcal{X}}_t$ , that is smaller than  $Z^\circ$ . In Fig. 6.9, we show an example of our sampling strategy, comparing uniform sampling with motion-driven sampling and, finally, our motion-and-feature-driven sampling. The picture shows how the proposed sampling procedure yields points on moving objects (chair), while also representing portions of the frame with different visual patterns. Differently, a uniform sampling ends up in giving a lot of emphasis to the empty regions of the image. In Fig. 6.8, the elements of  $\tilde{\mathcal{X}}_t$  are marked with white circles (toy example).

### Contrastive loss

We are now ready to setup a InfoNCE-like contrastive loss in the usual log-exponential form (see Sec. 2.3), giving more importance to positive pairs with large  $p_\delta$  and to negative pairs with large  $n_\delta$ ,

$$\begin{aligned} \mathcal{L}_+(g, h, \delta, p_\delta, n_\delta) = \\ - \sum_{x_i, x_j \in \tilde{\mathcal{X}}} \frac{p_\delta(x_i, x_j)}{Z} \log \frac{e^{s(x_i, x_j, \delta, g, h)}}{e^{s(x_i, x_j, \delta, g, h)} + \sum_{x_z \in \tilde{\mathcal{X}}} n_\delta(x_i, x_z) e^{s(x_i, x_z, \delta, g, h)}}, \end{aligned} \quad (6.10)$$

being  $Z = \sum_{x, y \in \tilde{\mathcal{X}}} p_\delta(x, y)$  the normalization factor. The outer summation is restricted to positive pairs, weighted by their degree of positiveness  $p_\delta$ . The first element of each positive pair,  $x_i$ , acts as an anchor, so that the similarity of the positive pair is contrasted by all the negative pairs involving  $x_i$  (weighted by  $n_\delta$ ), as can be appreciated by the denominator of the log-argument. Notice that the sums of Eq. 6.10 are restricted to the sampled coordinates, belonging to  $\tilde{\mathcal{X}}_t$ . Finally, our self-supervised objective  $L_{self}^\ell$  is obtained by instantiating Eq. 6.10 twice, relating features from the same frame (at  $t - 1$ )<sup>19</sup> and across two consecutive frames ( $t - 1$

<sup>17</sup>We use  $f_{t-1}$  to be consistent with the coordinate frame of  $\delta_t$ , and  $|a|$  is the element-wise absolute value of vector  $a$ .

<sup>18</sup>The overall number of sets ( $2d$ ) can be smaller in the case of one or more empty intersections (the normalization factor would be the number of non-empty sets instead of  $2d$ ).

<sup>19</sup>This is due to the fact that motion predictors output displacement vectors in the reference frame at  $t - 1$ .



and  $t$ ), respectively,

$$L_{self}^\ell = \mathcal{L}_+(f_{t-1}^\ell, f_{t-1}^\ell, 0, p_\delta, n_\delta) + \mathcal{L}_+(f_{t-1}^\ell, f_t^\ell, \delta_t^\ell, p_\delta, n_\delta). \quad (6.11)$$

leading to an in-frame contrastive objective and a cross-temporal contrastive objective, respectively. Notice that, in the first term of the summation, the flow is set to 0, since no warping is performed, as already discussed.

#### 6.5.4 Learning over time

Pairs of consecutive frames are processed in an online fashion, with a single-pass approach for each pair, and without any inputs from auxiliary memory buffers.

At each time step  $t$ , a new frame  $I_t$  is provided by  $\mathcal{V}$  and the previous one,  $I_{t-1}$ , is kept in cache. Learning consists in performing a single update of the model parameters in the whole network ( $\mathcal{F}$  and  $\mathcal{D}$ ) given frames  $I_{t-1}$  and  $I_t$ , with the goal of minimizing the total loss  $L$  that considers both Eq. 6.8 and Eq. 6.11,

$$L = \sum_{\ell} \left( L_{conj}^\ell(f_{t-1}^{\ell-1}, f_t^{\ell-1}, f_{t-1}^\ell, f_t^\ell, \delta_t^\ell, \delta_t^1) + L_{self}^\ell(f_{t-1}^\ell, f_t^\ell, \delta_t^\ell) \right), \quad (6.12)$$

where, for completeness, we also made explicit the list of arguments of  $L_{conj}^\ell$  and  $L_{self}^\ell$ . Such a loss function  $L$  is naturally regularized over time, due to the  $L_{conj}^\ell$  terms in the representation learning criteria, making it a natural instance of regularization-based approaches to continual learning. Moreover, the spatio-temporal bridge introduced on the sampled points by our contrastive loss (second term of the summation in Eq. 6.11), introduces further regularity over time. It is natural to wonder whether such regularity over time is enough to guarantee a good compromise between plasticity and stability of the networks, hence mitigating catastrophic forgetting, in both the motion estimators  $\mathcal{D}$  and the feature extractors  $\mathcal{F}$ .

We showed in Ch. 5 that, in general, the (regularized) criterion of Eq. 6.7 is enough to learn to estimate pixel displacements  $\delta$  over time by online gradient descent on neural parameters  $\gamma$ . Moreover, forgetting issues are not particularly evident due to the locality of the motion estimation process, resulting in negligible interference even in case of long streams with significant variability. On the opposite, our initial experimental findings indicated that learning over time poses significant challenges for the feature extractors  $\mathcal{F}$ , as we observed notable forgetting when approaching the problem using the naive approach. Then, we borrow an intuition from both contrastive learning (see Sec. 2.3) and continual learning, where there are two instances of the same model and some weights are updated by a momentum-based moving average. Despite being conceptualized in different configurations

(Tarvainen and Valpola, 2017; He et al., 2020; Arani et al., 2022), these approaches share the idea of using an Exponential Moving Average (EMA) scheme to update the parameters of a (teacher) network, while another network (student), which is continuously updated by gradient descent, is enforced to be consistent with it, preventing rapid changes in the parameter space. The EMA-updated network acts as a slowly progressing encoder, with parameters that evolve smoothly, and it can be employed both to stabilize learning of the student network and to better preserve information from the past.

Since our model is designed to extract and relate representations from a pair of frames  $(I_{t-1}, I_t)$ , we propose to extract features from the old frame using a feature extractor whose weights, referred to as  $\theta_t^{GRA}$  (GRAdient-updated), are updated by online gradient, while the current frame is processed by an EMA-updated network (slow learner) with the same architecture and weights  $\theta_t^{EMA}$ ,

$$\begin{aligned} f_{t-1}^\ell &= f^\ell(f_{t-1}^{\ell-1}, \theta_t^{GRA, \ell}) \\ f_t^\ell &= f^\ell(f_t^{\ell-1}, \theta_t^{EMA, \ell}), \end{aligned}$$

where, for the sake of clarity, we overloaded the notation, making explicit the weight argument. Fig. 6.10 summarizes the structure of whole model, emphasizing the two instances of the feature extractors (GRA and EMA). Such two networks are naturally bridged by the  $L_{self}$  (Eq. 6.11) component of the loss function  $L$ , that promotes coherence between them in the motion-driven manner proposed in this section. While the GRA network is updated by the gradient of  $L$  with respect to its weights and using learning rate  $\alpha_f > 0$ , the other network is updated by EMA with coefficient  $\zeta \in [0, 1)$ ,

$$\begin{aligned} \theta_{t+1}^{GRA} &= \theta_t^{GRA} - \alpha_f \nabla_{\theta} L(\mathcal{F}_{t-1}, \hat{\mathcal{F}}_t, \mathcal{D}_t) \\ \theta_{t+1}^{EMA} &= \zeta \theta_t^{EMA} + (1 - \zeta) \theta_{t+1}^{GRA}, \end{aligned}$$

where  $\nabla_{\theta} L$  is the gradient with respect to the weights in  $\mathcal{F}$  (the feature extractors), with  $\theta_1$  some random initialization of the neural parameters. Notice that  $\hat{\mathcal{F}}_t$  indicates that features  $\mathcal{F}_t$  are treated here as a constant value (they come from the EMA instances). The learning procedure is detailed in Alg. 5.

## 6.6 Experiments and discussion

**Setup.** To compare COAT and CMOSFET, we use the same experimental setup. The agent learns by watching the first 25 laps per object and, only in the subsequent laps, receives a total of 3 supervisions  $(k_t, y_t)$  per object, spaced out by at least 100 frames. Learning stops when all the objects complete 30 laps and, finally, performances are measured in the last lap, considering the F1 score (averaged over the

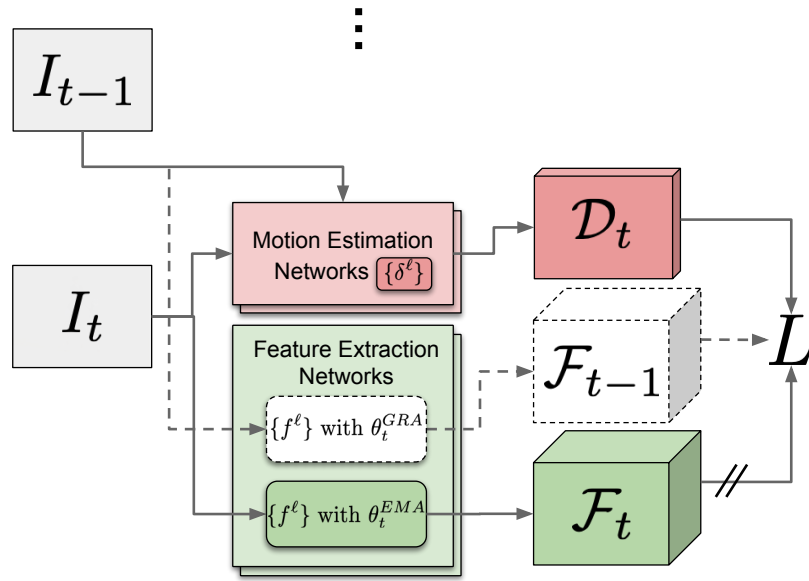


Figure 6.10: Learning over time with CMOSFET. Exponential Moving Average (EMA) networks and gradient-updated networks (GRA) extract features from  $I_t$  and  $I_{t-1}$ , respectively. White-dotted parts of the model are in addition to the ones of Fig. 6.6, and the diagonal bars ( $//$ ) indicate that no gradients are propagated on that path. The loss function  $L$  of Eq. 6.12 drives the learning process of the motion predictors and of the feature extractors, encouraging GRA to be coherent with EMA and instantiating the motion-induced contrastive criterion.

categories) associated with predictions of the open-set classifier<sup>20</sup>. Parameters of the attention model were adapted according to a preliminary run of the model ( $\alpha_m, \alpha_b, \mu$ ). For each video stream, we searched for the model hyper-parameters that maximize the F1 along the attention trajectory (1 pixel per frame), measured during the 30-th laps. To evaluate the performance of our approaches on the considered real-world videos, we adopted a very similar methodology. In this case, ground truth for evaluation comes from manual labelling, provided by Liang et al. (2020) on selected frames. The grids of parameter values that we considered and the optimal values are reported in Appendix B and C, together with further details.

**COAT: neural architectures** We evaluated the proposed approach considering two different families of deep convolutional architectures yielding  $d$  output features, referred to as RESUNET (UNet-like architecture, Ronneberger et al. (2015), based on ResNet) and FCN-ND (6-layer Fully-Convolutional without any downsamplings – Sherrah (2016)), respectively (see Appendix B for details). In this case, optical flow

<sup>20</sup>In the case of CMOSFET, features from all the levels  $\{f_t^\ell(x) | \ell = 1, \dots, N\}$  are considered and independently normalized for each  $\ell$ , in order to account for large differences in magnitude between the extractors.

---

**Algorithm 5** CMOSFET learning procedure. For each frame from stream  $\mathcal{V}$ , features are extracted, motion is estimated, and the proposed self-supervised loss  $L$  drives the learning processes. The GRA and EMA network instances are initialized to the same weight values.

---

**Require:** Stream  $\mathcal{V}$  of length  $T$ , potentially infinite; neural nets  $\mathcal{D}$  and  $\mathcal{F}$  with initial parameters set to  $\gamma_1$  and  $\theta_1$ . Learning rates  $\alpha_m, \alpha_f > 0$  and  $\zeta \in [0, 1)$ .

```

1:  $t \leftarrow 1, \theta_1^{GRA} \leftarrow \theta_1, \theta_1^{EMA} \leftarrow \theta_1$ 
2:  $f_0^0 = I_1, f_1^0 = I_1$ 
3:
4: while  $t \leq T$  do
5:    $I_t \leftarrow \text{next\_frame}(\mathcal{V})$ 
6:
7:   /* Cascade computation of flows and features */
8:   for  $\ell = 1, \dots, \ell = N$  do
9:      $\delta_t^\ell = \delta^\ell(f_t^{\ell-1}, f_{t-1}^{\ell-1}, \gamma_t^\ell)$            {▷ motion extraction on lower features}
10:     $f_{t-1}^\ell = f^\ell(f_{t-1}^{\ell-1}, \theta_t^{GRA, \ell})$            {▷ GRA instance is used}
11:     $f_t^\ell = f^\ell(f_t^{\ell-1}, \theta_t^{EMA, \ell})$            {▷ EMA instance is used}
12:   end for
13:
14:    $\mathcal{D}_t \leftarrow \{\delta_t^\ell | \ell = 1, 2, \dots, N\}$ 
15:    $\mathcal{F}_{t-1} \leftarrow \{f_{t-1}^\ell | \ell = 1, 2, \dots, N\}$ 
16:    $\mathcal{F}_t \leftarrow \{f_t^\ell | \ell = 1, 2, \dots, N\}$ 
17:
18:   /* Weight update equations */
19:    $\gamma_{t+1} \leftarrow \gamma_t - \alpha_m \nabla_{\gamma} L(\mathcal{F}_{t-1}, \mathcal{F}_t, \mathcal{D}_t)$    {▷ grad. motion estimator update}
20:    $\theta_{t+1}^{GRA} \leftarrow \theta_t^{GRA} - \alpha_f \nabla_{\theta} L(\mathcal{F}_{t-1}, \mathcal{F}_t, \mathcal{D}_t)$  {▷ grad. feature extractor update}
21:    $\theta_{t+1}^{EMA} \leftarrow \zeta \theta_t^{EMA} + (1 - \zeta) \theta_{t+1}^{GRA}$    {▷ EMA feature extractor update}
22:
23:    $t \leftarrow t + 1$ 
24: end while

```

---

is already available, since it is provided by the virtual environment (apart from the real-world videos, for which we used RAFT (Teed and Deng, 2020)). Learning to estimate the flow with a neural model (as we did in COAT) would just result in increased computational efforts, without adding anything to the comparison about the quality of the features<sup>21</sup>.

**CMOSFET: neural architectures.** In this case we have feature extractors  $\mathcal{F}$  and flow estimators  $\mathcal{D}$ . Each  $f^\ell$  and each  $\delta^\ell$  consists of a ResUNET architecture (similar to the one used by COAT), which is a variant of the U-Net network (Ronneberger

---

<sup>21</sup>Differently, in the case of CMOSFET, we have learnt optical flow estimation since it is crucial in the overall learning scheme.

Table 6.2: F1 scores over 10 runs (mean  $\pm$  std), in seven different video streams (EMPTYSPACE, SOLID, LIVINGROOM, RAT and HORSE with some -BW variants). The bottom part of the table is about the main competitors of the proposed model, including the raw-image (degenerate) baseline. The top part of the table collects reference results obtained with large-offline-pre-trained models, publicly available (references in Sec. 6.3).

	# Params	EMPTYSPACE		SOLID	LIVINGROOM		RAT	HORSE	
		BW	RGB	BW	BW	RGB	RGB	RGB	
OFFLINE PRE-TRAINED	DPT-C	121M	0.66	0.67	0.64	0.35	0.39	0.59	0.83
	DPT-B	120M	0.71	0.69	0.68	0.39	0.39	0.58	0.87
	DEEPLAB-C	58.6M	0.49	0.61	0.57	0.31	0.34	0.56	0.86
	DEEPLAB-B	42.5M	0.70	0.65	0.66	0.34	0.44	0.57	0.81
	MoCo v1	8.5M	0.73	0.73	0.74	0.33	0.35	0.70	0.66
	MoCo v2	8.5M	0.75	0.76	0.74	0.41	0.43	0.59	0.79
	MoCo v3	8.5M	0.76	0.76	0.76	0.41	0.44	0.58	0.64
	PIXPRO	8.5M	0.31	0.46	0.41	0.30	0.22	0.59	0.70
CONTINUAL	RAW IMAGE	-	0.50	0.45	0.18	0.10	0.23	0.67	0.66
	COAT (RESUNET)	17.8M	0.55 $\pm$ 0.03	0.71 $\pm$ 0.03	0.50 $\pm$ 0.01	0.31 $\pm$ 0.04	0.25 $\pm$ 0.07	0.59 $\pm$ 0.04	0.71 $\pm$ 0.07
	COAT (FCN-ND)	0.1M	0.60 $\pm$ 0.05	0.51 $\pm$ 0.07	0.48 $\pm$ 0.03	0.24 $\pm$ 0.01	0.28 $\pm$ 0.03	0.42 $\pm$ 0.05	0.50 $\pm$ 0.08
	CMOSFET-Single	1.1M	0.60 $\pm$ 0.06	0.78 $\pm$ 0.06	0.62 $\pm$ 0.02	0.33 $\pm$ 0.03	0.34 $\pm$ 0.05	0.61 $\pm$ 0.07	0.75 $\pm$ 0.06
	CMOSFET	2.3M	<b>0.64</b> $\pm$ 0.06	<b>0.82</b> $\pm$ 0.02	<b>0.64</b> $\pm$ 0.05	<b>0.38</b> $\pm$ 0.04	<b>0.36</b> $\pm$ 0.05	<b>0.74</b> $\pm$ 0.09	<b>0.84</b> $\pm$ 0.01

et al., 2015). We reduced the number of convolutional filters in each layer by a factor of 4, while the number of per-pixel outputs is 32 for  $f^\ell$  and 2 for  $\delta^\ell$  (horizontal and vertical flow components). We considered two implementations of this model: CMOSFET-Single, with a single level of feature and flow extractors (i.e.,  $N = 1$ ), and CMOSFET, with two levels, developing lower and higher-level motions (i.e.,  $N = 2$ ). For more details refer to Appendix C.1.

We apply a simple learning schedule to warm-up the motion estimators, avoiding premature training of the neural feature extractors. During the first  $T_{sched}$  laps we only learn the flow estimator at level 1. Then, if  $N > 1$ , we start learning the feature extractor on the same level, and we wait further  $T_{sched}$  laps before applying the same schedule on the following level. The procedure is repeated, progressively activating learning on the whole network (see Appendix C.2 for details).

### 6.6.1 Quantitative results

Table 6.2 reports the evaluation results of our proposed methods, considering the F1 scores over the entire frame. Specifically, we focus on the related *continual* self-supervised competitors (bottom part of the table), observing that CMOSFET outperforms COAT in all video streams (see Fig. 6.12 and Appendix C.3 for qualitative

comparison). The smaller variant CMOSFET-Single is already slightly superior to the competitors, except for `EMPTYSPACE-BW`, where it is on par with the best competitor. Although the superiority of the CMOSFET model is evident, we show in Fig. 6.12 (right) that COAT achieves higher F1 scores along the attention trajectory. We argue that the reason is its learning mechanism being biased towards the areas that tend to attract the human-like attention. Concerning COAT, we can see in the table that FCN-ND is often much less accurate, mostly due to the lack of spatial aggregation in its convolutional architecture (with respect to `RESUNET`).

Notably, the largest gap between CMOSFET-Single and CMOSFET is observed in real-world videos (`RAT`, `HORSE`) that confirms the importance of higher-level motion. We remark that here the flow estimation is learned online from scratch, in contrast to COAT, that relies on a flawless motion signal from the virtual environment. Moreover, CMOSFET models do not rely on explicit segmentation procedures for learning, yet they effectively exploits the relationships between flows and learned features. Comparing `EMPTYSPACE-BW` and `EMPTYSPACE-RGB`, we notice a significant gap in F1, suggesting that even though the `RAW` scores of grayscale and color representations are similar, both COAT and CMOSFET efficiently exploits the intrinsically richer information encoded in RGB. Additionally, we observe that CMOSFET encodes pixels in a reduced number of output features compared to competitors ( $32 \cdot 2$  vs. up-to-128), and has significantly fewer learnable parameters, with only 2M parameters (compared to 18M of `COAT-RESUNET`, Tab. 6.2). This confirms the capability of our method to develop informed but compact representations. Surprisingly, our approach achieves interesting performance even when compared to offline-pre-trained large architectures with 10-100M parameters (upper part of the table). It is important to note that our goal is not to overcome large-scale massively offline-trained models, since we learn from scratch in an online manner, specializing in the target environment. Nonetheless, CMOSFET beats these competitors in `EMPTYSPACE-RGB` and `RAT-RGB`. We highlight the performance on the other real-world video stream, `HORSE-RGB`, where CMOSFET overcomes most of the offline-trained models (all of the self-supervised ones) and it is only slightly behind the best one. These results affirm the adaptability of our approach to the properties of the environment at hand, with a fraction of the learnable parameters and processed data.

### 6.6.2 Qualitative results

Fig. 6.11 presents a qualitative analysis of the outputs of our models on three sample frames, with similar trends observed across all video streams. Specifically, the second and third columns depict the extracted flows from CMOSFET's first and second levels, respectively. The features produced by CMOSFET allow the classification procedure (recall that such procedure do not affect the feature learning process) to completely discriminate all the different target objects (fourth column, predic-

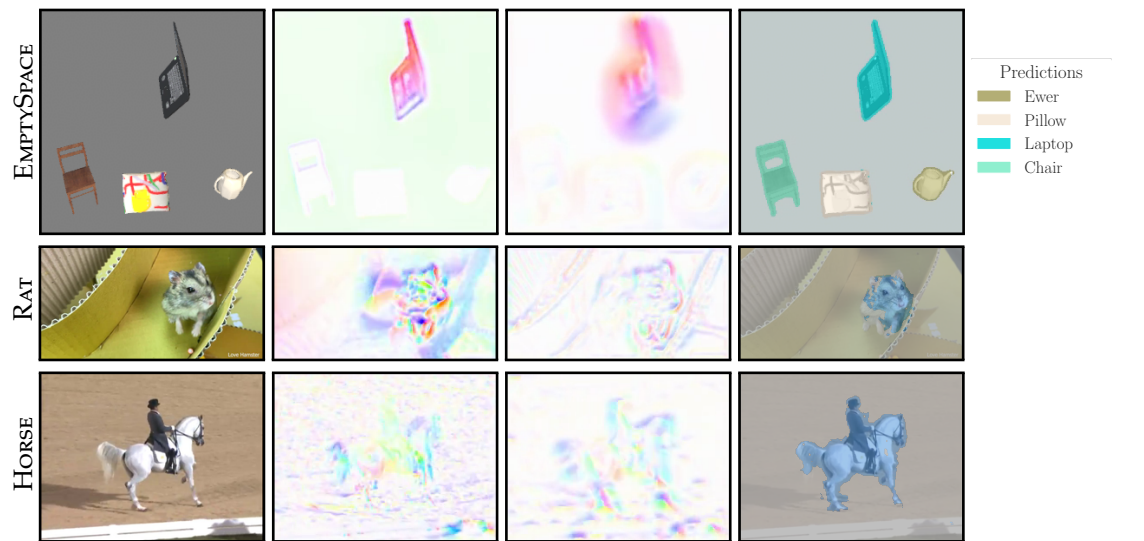


Figure 6.11: Three frames sampled from EMPTYSPACE, RAT and HORSE streams (first column). CMOSFET flows (first-level and higher-level in second and third columns, respectively) are plotted according to the optical flow conventional color mapping—see the color wheel on the right (Baker et al., 2007). Predictions in the fourth column show a pretty satisfying adherence to the borders of the objects (target objects are highlighted with colors; the background class is present in *all* the streams, reported with a light gray overlay).

tions). As it can be seen, the borders of the CMOSFET-based predictions appear slightly thicker compared to the ground-truth object borders. Indeed, contrastive learning is affected by noisy flow (recall that it is learned from scratch, jointly with the features), which can easily result in *false* positive pairs, with one pixel belonging to an object and the other just outside of it. Noticeably, the estimated optical flow (second column) is very crisp in the case of EMPTYSPACE where the camera is fixed, while it captures the effects of the moving-camera in real-world streams (second and third row). Also, optical flow estimation is known to be inherently challenging in nearly texture-free surfaces. Interestingly, comparing the two flow estimators of distinct levels (second and third column), we notice how they appear to focus on different parts of the frames, with flows that are somewhat related but clearly not overlapping. This confirms that the network develop a *latent* higher-level motion, challenging to interpret, but well suited to the learning process. It is worth noting that, consistently with Ch. 5, we did not use any specific trick (Zhai et al., 2021) to improve the flow extraction, such as spatial pyramids or occlusion handling, for the sake of simplicity.

We investigated our results, showing in Fig. 6.12 (left) examples of pixel-wise classification, highlighting some failures of Transformer-based DPT-B compared to COAT and CMOSFET. Interestingly, using features from the considered state-of-the-

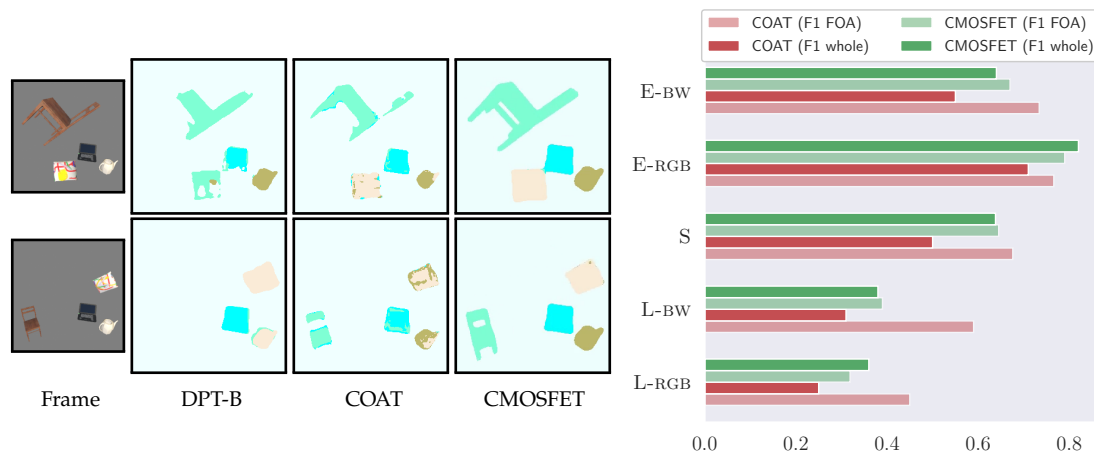


Figure 6.12: Left: pixel-wise classification in two sample frames, comparing COAT (RESUNET) and CMOSFET with DPT-B (Transformers) on EMPTYSPACE stream. Different colors indicate different class predictions. Right: comparison between F1 measured on the whole frame and F1 measured on the focus of attention (FOA). While in Tab. 6.2 it is evident that CMOSFET attains higher performance in general, in this barplot we can appreciate that COAT (RESUNET) achieves better F1 along the attention trajectory.

art models can result in troubles in recognizing closer objects and also in discriminating objects from the background, due to the fact that their pixel representations are typically strongly affected by a large context (i.e., such features do not transfer well to our task). When presenting objects in unusual orientations, likely different from what observed during the fully supervised training, they tend to perform badly. Differently, our models—especially CMOSFET—adapt to the video stream, learning coherent representations, that are useful when the object transforms its appearance. Concerning COAT, the attention model focuses the learning process on what is more important, and, although just a tiny number of supervisions are provided, our model can roughly distinguish object, sometimes better than large offline-trained competitors.

### 6.6.3 COAT: ablation studies

In order to evaluate the sensitivity of COAT approach to the key elements of the considered setup, we selected the RESUNET model of Table 6.2 and evaluated the F1 on the focus of attention (the most significant one for this approach, as we remarked in Sec. 6.6.1), under different conditions. Fig. 6.13 reports results of experiments in which we changed the number of supervisions per object (Sec. 6.3), the number of edges  $e$  (per type) in the stochastic graph (Sec. 6.4.6), we disabled the temporal coherence (Sec. 6.4.3), and we changed the length of the streams (discarding SOLID in which differences were less appreciable). Even with a single supervision,



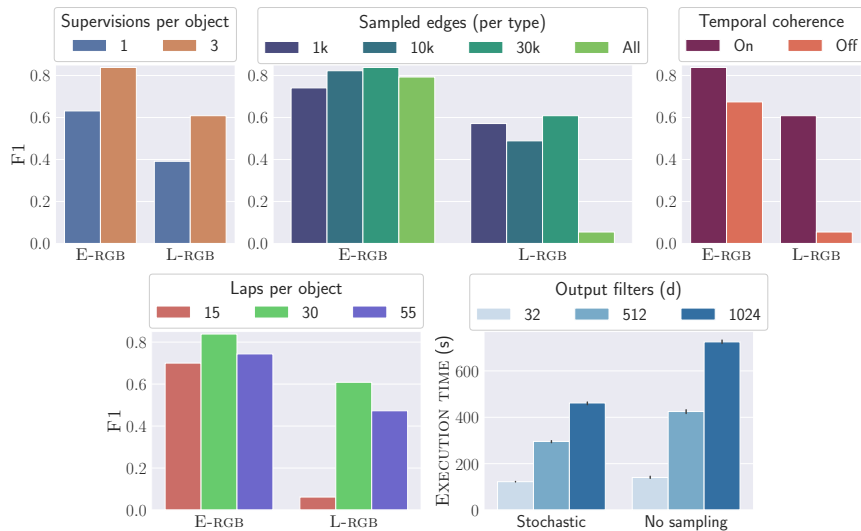


Figure 6.13: First 4 plots (left-to-right, top-to-bottom): in-depth COAT experiments on EMPTYSPACE (E) and LIVINGROOM (L) RGB streams (F1 FOA—on focus of attention trajectory). Last plot: timings (3 runs) for stochastic sampling ( $e = 10k$ ) and no subsampling at all, varying  $d$ .

the model is able to distinguish the target objects in EMPTYSPACE, while in the more cluttered LIVINGROOM it benefits from multiple supervisions, as expected. Our proposal is better than using a non-stochastic criterion (LIVINGROOM), and works well even with a limited number  $e$  of edges. Moreover, the agent benefits from relatively longer streams (going from 15 to 30 laps), that allow it to develop more consistent representations, and temporal coherence has an important role in the overall results. For completeness, we highlighted in Fig. 6.13 (last) the computational benefits, in terms of processing time (one lap per object,  $\approx 1.5k$  frames, i/o time is included), brought by the stochastic subsampling, showing that they are more evident—with respect to the total computational burden—for large  $d$ .

#### 6.6.4 CMOSFET: ablation studies

In order to better understand the impact of key design choices, we perform ablation experiments concerning some of the main components of CMOSFET, reporting our findings in Fig. 6.14, that we describe from left to right, top to bottom. The motion-feature conjugation term  $L_{conj}$  from the loss function (Eq. 6.8) provides improvements in all the streams (1st barplot), when compared to the case in which feature extractors are only driven by the contrastive loss (i.e., Eq. 6.8 is composed of *iii.* only), more evidently in real-world streams. It also improves stability among the different runs, resulting in reduced standard deviation—apart from RAT, where the lack of full conjugation leads to collapse to a very suboptimal solution with minimal variance. The adoption of a specific sampling strategy (2nd barplot) for the

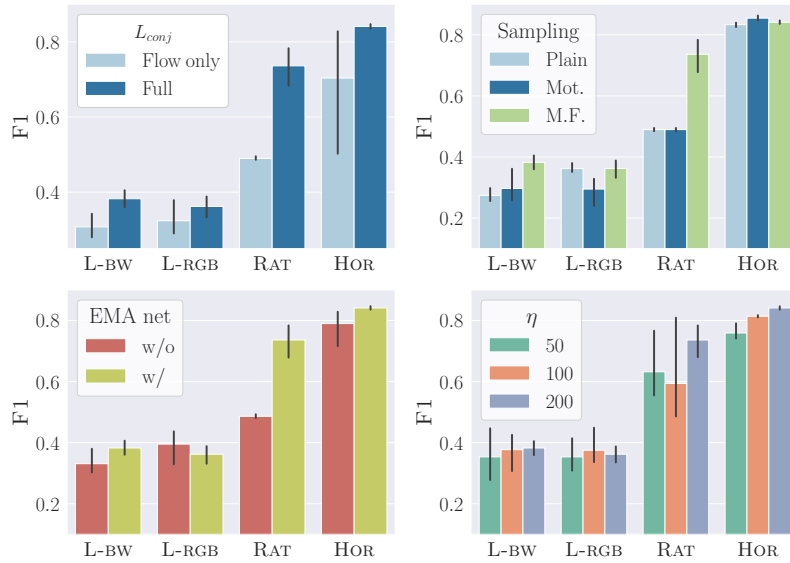


Figure 6.14: Ablation of CMOSFET components on four visually-rich streams (L stands for LIVINGROOM, HOR for HORSE). F1 is computed over the whole frame. From left to right, top to bottom: the impact of the features-motion conjugation term  $L_{conj}$  as presented in Eq. 6.8 (Full) compared to the case in which features are not driven by the conjugation loss (i.e., when Eq. 6.8 is composed of *iii.* only–Flow only); the role of different sampling strategies (Plain, Motion–Mot., Motion and Features–M.F.); EMA network (without or with); number of sampled points  $\eta$ . See Sec. 6.5 for details.

contrastive term, guided by both Motion and Features (M.F. in Fig. 6.14), also had a considerable influence on the results, sometimes dramatic as in the case of RAT. In general, we notice that different sampling strategy (Sec. 6.5.3) might be optimal in different streams, in relation with the kind of motion that is present in the respective environments. Motion-only-guided sampling (Mot.) can yield slight improvements with respect to vanilla uniform sampling strategy (Plain), due to the fact that it helps focusing on the rather small moving areas. Learning without the stabilizing effect of the EMA network (Sec. 6.5.4) generally leads to lower performance (3rd barplot), in agreement with literature in both contrastive learning and continual learning. The learning process is however still effective (apart from the challenging RAT), thanks to regularization effects introduced by the spatio-temporal loss terms, although less stable than when using the EMA scheme. We also separately evaluated the impact of the number of sampled locations  $\eta$  in the contrastive objective (Sec. 6.5.3). It turns out (4th barplot) that models developed in real-world streams consistently benefit from a larger number of locations ( $\eta = 200$ ). Conversely, this trend is not evident in synthetic streams, that have large static/uniform areas, where more dense sampling does not yield more information.

### 6.6.5 Remarks

In this chapter, we faced the challenge of developing an agent that learns to predict pixel-wise identity information in a fully unsupervised continual manner (open-set class-incremental setting). We presented novel approaches, leveraging focus of attention, spatio-temporal consistency, and motion-based contrastive objectives.

Concerning our most promising method, CMOSFET, we considered multi-level features and multi-level motion flows as *conjugated* one to each other, mutually constraining their development. CMOSFET is implemented by means of neural models trained to extract features and flows from scratch, driven by multi-level spatio-temporal feature-motion constraints paired with a self-supervised contrastive loss that promotes the emergence of meaningful features. We performed experiments on video streams from virtual environments and real-world videos, showing how CMOSFET benefits from learning multi-level flows, significantly overcoming COAT. Notably, CMOSFET performance are comparable to those of state-of-the art larger models, obtained with large-scale offline training.

# Chapter 7

## Conclusions and future works

In this chapter, we summarize findings of the work presented in this thesis and we highlight open issues for future research.

### Progressively learning in the input space

In the first part of this thesis we suggested techniques to improve the training of neural models, based on the idea of emulating natural gradual learning processes by altering visual stimuli in the input space (input tuning).

Friendly Training is a technique to automatically break down a single, complex learning problem into a sequence of small subtasks, each incrementally more challenging than the last. This learning schedule is actually implemented through input transformations (simplifications), tuning the cognitive complexity of each step to match the learner’s current skill level. We proposed two implementations differing in the way the tuned input is obtained, one based on direct-optimization and the other involving an auxiliary neural model. Assessing the performance improvement on a variety of benchmarks, we showed that the neural-based implementation is largely preferable. This methodology not only enhances the model’s performance but also fosters a more nuanced understanding of the learning process.

After a neural network has been trained on some well-defined task associated with a certain data collection, it is often required to adapt on novel data, sometimes incurring in non-stationary learning environments. Continual Input Tuning is designed exactly for this scenario, plagued by the well-known catastrophic forgetting problem. In this approach, we froze the model’s backbone and we exploited learnable transformations in the input space for environmental adaptation. The proposed method is simple but effective in consistently improving the quality of the outcome on multiple benchmarks.

**Learning plans and robustness.** In both implementations of Friendly Training, a set of parameters plays a pivotal role in governing the degree of simplification, indirectly controlling the distance between the original sample and the transformed counterpart. We have opted for quadratic laws to describe the evolution of these parameters, meaning that the learning schedule is intense at the beginning, avoiding prolonged engagement with overly simplified data, with barely no information. Subsequently, the learning pace is more gradual, to allow for effective learning of all the non-trivial complex details that should be captured in the input data. While our results demonstrate notable performance enhancements, it remains plausible that alternative learning strategies may yield higher levels of accuracy. We think that an interesting direction for future works lies in a proper investigation of different learning plans, leading to different trajectories in the weight space. Also, we conjecture that the proposed methodology can help in increasing the robustness to adversarial attacks and forgetting, since it can discourage the discovery of spurious correlations, but exhaustive experiments are needed.

**Theoretical understanding of forgetting.** Continual Input Tuning allows the network to effectively leverage the pre-training knowledge and find a good trade-off between plasticity and stability with modest computational efforts. However, the reason for this success is not entirely clear. While the improvements can be intuitively connected with findings about the prevalence of forgetting in the last layers of the network (Lesort et al., 2021; Ramasesh et al., 2021), it is of utmost importance to gain deeper insights into learning dynamics of neural networks when exposed to non-stationary environments. For this reason, we plan to get deeper insights into the evolution of our learnable transformations over time.

Furthermore, Continual Input Tuning has been devised as a proposal for the challenges posed by traditional continual learning benchmarks. Nonetheless, we think that it would be interesting to explore the adaptation of this strategy in contexts where the learning sequence is defined by a meaningful temporal evolution. We contend that, in such scenarios, incorporating concepts from Friendly Training could yield an even more compelling analogy and potentially enhance the efficacy of the approach.

## Growing visual agents from video streams

In the second part of this thesis we proposed two methods (COAT and CMOSFET) for self-supervised online learning on video streams, designed with the broad aim of contributing to the development of autonomous visual agents. We proved that the proposed methods lead to features that are useful in distinguishing objects, with the more recent method (CMOSFET) exhibiting significantly better performance. For

evaluation, we adopt an open-set class-incremental setting, exploiting 3D Virtual Environments, as well as real-world video streams. Results are, on average, comparable or better than the ones obtained using state-of-the-art models, pretrained on large visual collections, confirming the interest of these proposals for the research community.

**Improving stability.** Recent works highlighted the inherent unstable nature of self-supervised contrastive criteria (Zhang et al., 2021). CMOSFET is a pixel-wise motion-based contrastive criterion, proposed in the context of a completely online learning procedure, and as such there are some stability issues, confirmed by the non-negligible standard deviation of our results over multiple different initializations. Another intrinsic source of instability stands on the sensitivity to the effects of the first learning steps, where the flow estimators are obviously far from being able to produce reasonable flows, thereby negatively affecting the feature development. We would like to investigate whether introducing a developmental plan along the lines of Friendly Training (Ch. 3) can help in increasing the stability. This would involve gradually increasing the perceptual complexity of the video stream (resolution, texture, scene composition, etc.) or gradually increasing the complexity of the contrastive objective.

**Dealing with moving cameras.** COAT and CMOSFET are designed to learn in environments where background areas are characterized by motion patterns that are different from the ones of objects to which semantics are expected to be attached (or that are almost static). Our results show that small (or relatively slow) camera motion, as the one of the selected real-world videos, can be easily handled or filtered out, as we did in our implementation. However, we would like to develop specific countermeasures to deal with strongly moving cameras. This would involve the development of techniques for camera motion estimation and compensation, possibly learnt together with the other neural modules in a fully end-to-end model.

**Continual strategies.** Concerning the *continual* aspect of the learning problem, we have chosen a minimalistic online approach. However, we do not have extensive evidence concerning performance with extremely long streams (e.g., more than 24 hours) and large number of object categories (e.g., hundreds of categories). We argue that more advanced continual learning strategies could be useful in mitigating forgetting issues in such challenging situations. For instance, it could be useful to heuristically detect steady states in the overall evolution of the system, in order to take model snapshots and enforce consistency on extremely long timescales. Another direction would be the investigation of neural networks with architectures especially suited for continual learning problems (cfr. ongoing work—(Tiezzi et al.,

2023a)—in our research group, concerning the introduction of memory-augmented neurons).

**Improving perception and interaction.** We introduced the idea of simulating the trajectory of human-like attention, both for efficiently referencing pixel-wise supervisions and for learning objectives (COAT, in particular). It would be interesting to consequently change the perception model of the visual agent, by means of foveated neural models (see our recent work, not included in this thesis—(Tiezzi et al., 2023b)). Foveated computation allows location-dependent convolutions, i.e., fine-grained processing in a given-focused area and coarser processing in the peripheral regions. This would enable shifting the understanding of visual scenes from static frame processing to dynamic scene parsing along the trajectory of the focus of attention, fostering self-discovery of objects and more natural linguistic interaction with the human supervisor.

## Unifying perspective

In the Introduction of this thesis we proposed a visionary goal for research in Machine Learning and Computer Vision. We advocate for a new paradigm, where machines are free to learn continuously from their surroundings, reducing their reliance on extensive pre-existing datasets and seeking occasional human guidance. Drawing an analogy to human development, we noticed that both fundamental skills (such as visual scene understanding) and more abstract cognitive skills (such as the ones obtained through education) are not instantly transferred to newborns but undergo smooth development through time. While the complete realization of this paradigm shift is undoubtedly ambitious, these general arguments led to the development of specific input tuning techniques, presented in the first part of this thesis. These techniques are grounded in the context of traditional supervised machine learning, but they were conceived around the idea of modulating the information flow by altering visual stimuli in the original input space.

In the second part of the thesis, we pursued the broader objective of unsupervised visual representation learning, albeit within a controlled framework. As we posited several times, the need for extensive data collections in machine learning may arise from an unnecessarily complex problem formulation, that deviates from the natural learning process. To address this, we introduced two methods that autonomously extract semantic information from exposure to video streams with moving objects—an essential skill for realizing our long-term goal of autonomous visual agents.

It is important to note that further research is required to attain a satisfying level of robustness for real-world applications, which represents a longer-term endeavor.

We believe that the insights gleaned from our experience with input tuning techniques, particularly in controlling the learning pace, can significantly contribute to achieving greater stability and robustness in self-supervised learning methods for autonomous visual agents.

From a broader perspective, agents that continuously learn can have an extremely profound technological impact. However, the concept of continuously evolving autonomous systems can elicit concerns among the general public due to misconceptions and misunderstandings. We assert that the work discussed in this thesis, along with potential future developments, poses no inherent risk of negative societal consequences. Moreover, in the long run, our approach has the potential to address privacy and security concerns, since agents can excel in their tasks without relying on models from uncontrolled sources, operating in private target environments.



# Appendix A

## Experimental details on optical flow estimation

This supplemental section includes more details about some aspects of continual learning of optical flow estimation, presented in Ch. 5. Code, instructions, hyperparameters and 3D scenes<sup>1</sup> are publicly available<sup>2</sup>.

### A.1 Detailed Results

#### A.1.1 Standard deviations

Throughout Ch. 5 we always reported average numerical results over three runs with different seeds. For the sake of completeness, in Tab. A.1, A.2 we report the complete results shown in Tab. 5.1, 5.2 enriched with the standard deviations over the three runs. We chose to neglect this information in the main discussion, since it is always quite small and not very informative.

Table A.1: RECONSTRUCTION ACCURACY (%) on the considered streams/settings. We report the best model between the one that always updates its parameters (ALWAYS) and Div update policy (adding a \* when Div is reported). Best results among the models considered in the proposed experience are in bold.

Model	LEARNING				FROZEN			
	A	B	C	M	A	B	C	M
RESUNET	86.7 ± 0.4	<b>76.8</b> ± 0.2	92.3 ± 0.1	85.6 ± 0.2	84.0 ± 0.4	<b>78.6</b> ± 0.9 *	90.0 ± 0.8 *	86.1 ± 0.0
ND CONV	<b>87.4</b> ± 0.3	76.3 ± 0.2	92.9 ± 0.1	<b>87.3</b> ± 0.0	86.0 ± 0.2	77.5 ± 0.1	91.2 ± 0.4	<b>87.9</b> ± 0.1
DN CONV	86.0 ± 0.2	76.3 ± 0.6*	93.0 ± 0.1	86.4 ± 0.0	<b>86.2</b> ± 0.2	77.3 ± 0.2 *	91.1 ± 0.1	86.5 ± 0.1 *
FLOWNETS	83.7 ± 0.1	73.4 ± 0.3*	<b>93.1</b> ± 0.0	82.7 ± 0.1	81.7 ± 0.1 *	69.3 ± 3.0 *	<b>91.8</b> ± 0.2	83.3 ± 0.1

<sup>1</sup>TDW software (v1.9.1) is needed. Available at <https://www.threedworld.org/>

<sup>2</sup><https://github.com/sailab-code/continual-of>

Table A.2: MOTION-F1 in fixed-camera streams. The \* has the same meaning as in Tab. A.1.

Model	LEARNING		FROZEN	
	A	B	A	B
RESUNET	$0.829 \pm 0.008$	$0.661 \pm 0.002$ *	$0.834 \pm 0.012$ *	$0.712 \pm 0.031$ *
NDCONV	<b><math>0.840 \pm 0.006</math></b>	$0.667 \pm 0.007$	$0.818 \pm 0.003$ *	<b><math>0.738 \pm 0.015</math> *</b>
DNDCONV	$0.836 \pm 0.003$	<b><math>0.682 \pm 0.020</math> *</b>	$0.827 \pm 0.006$	$0.701 \pm 0.004$ *
FLOWNETS	$0.771 \pm 0.002$ *	$0.623 \pm 0.009$ *	$0.784 \pm 0.001$ *	$0.425 \pm 0.253$ *

### A.1.2 Forgetting

In this section we deepen our analysis on the *forgetting* issue, previously discussed (see Fig. 5.4, 5.5), where we investigate the impact of forgetting earlier experience by freezing the weights. We show the performance obtained leveraging the different proposed update policies (see Sec. 5.3.1) and the alternative of updating the weights on every new frame (ALWAYS policy). We show the performance via the MOTION-F1 metric (in Tab. A.3) and the RECONSTRUCTION ACCURACY (Tab. A.4) metric. We report the results obtained on all the streams and in both the “learning” and “frozen” settings. Additionally, we provide measurements restricted to the first (long-term – LT) and the last (short-term – ST) 1-minute windows (notice that ST and LT have been visualized along the axes of the scatter plots in Fig. 5.5). The positive impact of the update policies is mostly visible in the “frozen” setting. By looking at the relatively small difference between columns LT and ST, we notice that *catastrophic* forgetting does not seem to be an actual issue for the specific task of Optical Flow estimation.

Table A.4: RECONSTRUCTION ACCURACY. Subcolumns as defined in Tab. A.3.

Policy	STREAM A				STREAM B				STREAM C				STREAM M				
	L	F	ST	LT	L	F	ST	LT	L	F	ST	LT	L	F	ST	LT	
RESUNET	ALWAYS	86.7	84.0	83.6	84.5	76.8	76.7	72.8	73.3	92.3	89.8	90.4	88.8	85.6	86.1	95.0	91.2
	DEC (50)	65.4	65.7	55.0	66.1	67.3	69.5	66.2	67.8	90.4	90.5	90.2	89.3	75.1	76.0	89.1	84.7
	DIV (0.05)	84.7	83.6	83.0	84.0	76.8	78.6	74.6	75.1	90.9	90.0	90.1	89.0	85.1	85.8	94.8	91.0
NDCONV	ALWAYS	87.4	86.0	85.3	86.2	76.3	77.5	72.0	73.3	92.9	91.2	91.3	90.4	87.3	87.9	95.8	92.6
	DEC (50)	74.1	81.5	79.7	81.8	68.0	70.5	66.3	68.3	90.5	90.5	90.3	89.4	74.5	74.5	87.5	83.4
	DIV (0.05)	86.0	85.7	84.3	86.3	76.1	77.0	72.1	73.1	92.2	89.9	90.6	89.1	86.9	87.7	95.5	92.3
DNDCONV	ALWAYS	86.0	86.2	85.1	86.5	75.1	76.7	71.3	72.5	93.0	91.1	91.5	90.4	86.4	86.3	94.8	91.5
	DEC (50)	65.6	69.1	58.0	69.9	67.2	68.2	65.0	66.2	90.5	90.5	90.2	89.3	74.4	74.5	87.6	83.4
	DIV (0.05)	84.7	85.9	84.9	86.2	76.3	77.3	71.8	72.9	92.2	90.4	90.8	89.7	86.2	86.5	94.9	91.5
FLOWNETS	ALWAYS	83.7	81.2	81.2	81.3	73.2	66.4	64.1	65.5	93.1	91.8	92.3	90.8	82.7	83.3	92.7	88.6
	DEC (50)	64.7	65.0	54.1	65.5	65.8	67.7	64.1	66.2	90.0	90.2	90.1	88.9	74.2	74.4	87.3	83.4
	DIV (0.05)	83.1	81.7	81.3	81.5	73.4	69.3	64.8	67.1	92.4	91.4	92.0	90.3	82.4	82.9	92.6	88.4

Table A.3: MOTION-F1 in fixed-camera streams (streams A and B) in different settings: learning (L), frozen all-stream (F), frozen short-term trial (ST), frozen long-term trial (LT). We compare DEC with  $n = 50$ , DIV with  $l = 0.05$  and ALWAYS policy.

Policy		STREAM A				STREAM B			
		L	F	ST	LT	L	F	ST	LT
RESUNET	ALWAYS	0.829	0.803	0.869	0.792	0.659	0.618	0.540	0.497
	DEC (50)	0.013	0.024	0.022	0.024	0.114	0.324	0.251	0.161
	DIV (0.05)	0.821	0.834	0.878	0.825	0.661	0.712	0.599	0.601
NDCONV	ALWAYS	0.840	0.807	0.884	0.801	0.667	0.718	0.678	0.601
	DEC (50)	0.479	0.699	0.816	0.704	0.235	0.332	0.287	0.307
	DIV (0.05)	0.833	0.818	0.886	0.806	0.663	0.738	0.659	0.592
DNDCONV	ALWAYS	0.836	0.827	0.879	0.822	0.643	0.715	0.699	0.650
	DEC (50)	0.030	0.237	0.250	0.232	0.196	0.700	0.355	0.339
	DIV (0.05)	0.822	0.823	0.878	0.816	0.682	0.717	0.701	0.686
FLOWNETS	ALWAYS	0.768	0.764	0.837	0.751	0.618	0.059	0.076	0.027
	DEC (50)	0.013	0.001	0.000	0.002	0.027	0.198	0.175	0.086
	DIV (0.05)	0.771	0.784	0.866	0.773	0.623	0.425	0.362	0.237

### A.1.3 Data biases

In the main discussion we have also presented experiments designed to investigate the effect of specific biases in the input stream. In particular, in Fig. 5.6 (stream A) we showed what happens when playback and pause are interleaved in order to simulate static segments of real-world streams. In Tab. A.5, A.6 we report results in “learning” and “frozen” settings on all the streams and the architectures.

### A.1.4 Longer streams

One could be interested in quantitatively assessing the impact of long-stream exposure. For this reason, we provide in Tab. A.8 numerical results (running averages) for the stream concatenation experiment (see Ch. 5, denoted with ABCM), considering the ALWAYS update policy and the proposed alternatives (DIFF, MAG, DIV). Fig. 5.7 in Ch. 5 includes visualizations of the evolution of the metrics when considering the RESUNET model. Such information can be found, aggregated over time, in top rows of Tab. A.8.

A different experiment (Tab. A.7) concerns updating the weights on a second exposure to the streams, that significantly improves the metrics (overcoming the baseline on two streams out of three).

Table A.5: MOTION-F1 when injecting static shots in fixed-camera streams, varying the update policy. Learning (L), Frozen (F). We compare DIFF with  $q = 0.001$ , MAG with  $r = 0.02$  and  $h = 0.2$ , DIV with  $l = 0.05$  and ALWAYS policy.

Policy		STREAM A		STREAM B	
		L	F	L	F
RESUNET	ALWAYS	0.788	0.176	0.596	0.447
	DIFF	0.679	0.690	0.588	0.651
	MAG	0.797	0.663	0.618	0.624
	DIV	0.784	0.207	0.634	0.486
NDCONV	ALWAYS	0.820	0.541	0.646	0.521
	DIFF	0.788	0.714	0.629	0.640
	MAG	0.795	0.745	0.640	0.644
	DIV	0.769	0.462	0.650	0.483
DNDCONV	ALWAYS	0.781	0.443	0.655	0.588
	DIFF	0.764	0.739	0.627	0.658
	MAG	0.793	0.729	0.659	0.666
	DIV	0.755	0.672	0.655	0.617
FLOWNETS	ALWAYS	0.730	0.751	0.612	0.496
	DIFF	0.730	0.788	0.569	0.609
	MAG	0.730	0.792	0.616	0.641
	DIV	0.729	0.711	0.612	0.187

### A.1.5 Comparison with Horn-Schunck algorithm

In Ch. 5 we presented a comparison between the proposed neural-based approach and classic iterative algorithms, i.e., Horn and Schunck (1981). In Tab. A.9 we show that the MOTION-F1 obtained in the case of the HS algorithm is much lower compared with our proposal, to the point that it is unlikely to be useful for downstream applications. For the sake of completeness, in the same table we show the RECONSTRUCTION ACCURACY. This apparently good result highlights that achieving good performance in both the metrics is crucial for a satisfactory solution. To emphasize the capability of the neural models to improve over time, we consider the “learning” setting when the stream is shown a 2nd time (LEARNING+) and the “frozen” setting after having learnt from ABCM (FROZEN+).

Table A.6: RECONSTRUCTION ACCURACY (%) when injecting static shots in the streams, varying the update policy. Learning (L), Frozen (F). See Tab. A.5 for details.

	Policy	STREAM A		STREAM B		STREAM C		STREAM M	
		L	F	L	F	L	F	L	F
RESUNET	ALWAYS	92.1	84.1	85.8	84.0	95.9	95.4	90.9	91.9
	DIFF	92.0	90.5	86.2	86.3	95.1	94.4	88.9	92.1
	MAG	92.2	89.6	86.1	86.7	95.5	95.3	91.0	91.8
	DIV	91.5	84.3	86.4	85.0	94.8	95.1	88.5	91.5
NDCONV	ALWAYS	92.8	88.6	86.1	85.6	96.2	95.3	92.1	92.4
	DIFF	92.9	91.8	86.2	87.2	95.6	92.3	91.1	92.3
	MAG	92.5	91.5	86.0	87.3	96.0	95.5	92.1	92.2
	DIV	91.5	87.3	86.3	84.9	95.5	95.5	92.1	92.1
DNDCONV	ALWAYS	91.6	87.2	86.1	85.9	96.0	95.3	91.8	91.8
	DIFF	91.8	91.7	85.7	86.6	95.5	93.9	90.0	92.0
	MAG	91.8	91.2	86.1	87.3	95.5	95.4	91.8	91.7
	DIV	90.7	90.3	86.2	86.8	95.3	95.4	91.6	91.7
FLOWNETS	ALWAYS	90.8	89.4	85.4	83.9	96.3	95.4	89.3	89.6
	DIFF	90.8	90.9	85.1	85.0	95.4	94.1	87.7	89.1
	MAG	90.8	91.0	85.4	85.5	96.0	95.5	89.2	88.9
	DIV	90.5	89.0	85.3	82.7	95.4	95.1	88.5	88.7

Table A.7: RECONSTRUCTION ACCURACY (left) and MOTION-F1 (right) on the shorter streams, “learning” setting, presenting the stream a second time.

	Model	A	B	C		Model	A	B
REFERENCE	RESUNET	0.902	<b>0.805</b>	0.934		RESUNET	0.858	0.707
	NDCONV	<b>0.907</b>	0.790	0.944		NDCONV	<b>0.872</b>	0.693
	DNDCONV	0.895	0.788	<b>0.947</b>		DNDCONV	0.869	<b>0.715</b>
	FLOWNETS	0.863	0.745	0.939		FLOWNETS	0.800	0.638
	NULL	0.652	0.659	0.905				
	FLOWNETS	0.762	0.693	0.914				
	RAFT-SMALL	0.839	0.739	0.946				
	SMURF	0.876	0.798	0.958				
	RAFT	0.859	0.772	0.954				

Table A.8: MOTION-F1 (left) and RECONSTRUCTION ACCURACY (right) when streams are sequentially shown to the learning agent (A, B, C, M). We compare DEC with  $n = 150$ , Div with  $l = 0.05$  and ALWAYS policy.

Policy		LEARNING+		FROZEN+		Policy		LEARNING+				FROZEN+			
		A	B	A	B			A	B	C	M	A	B	C	M
RESUNET	ALWAYS	0.824	0.709	0.813	0.664	RESUNET	ALWAYS	86.4	79.1	95.5	88.0	89.2	81.7	95.2	87.7
	DEC (150)	0.000	0.667	0.787	0.664	RESUNET	DEC (150)	65.1	76.0	95.2	87.7	88.8	81.0	95.0	87.1
	Div (0.05)	0.823	0.702	0.807	0.672	RESUNET	Div (0.05)	85.2	78.2	94.9	87.7	88.8	81.0	95.0	87.3
NDCONV	ALWAYS	0.848	0.689	0.848	0.633	NDCONV	ALWAYS	87.7	79.0	95.4	87.8	88.9	80.5	95.2	86.9
	DEC (150)	0.069	0.682	0.851	0.630	NDCONV	DEC (150)	66.0	76.9	95.2	87.5	88.5	79.7	95.0	86.6
	Div (0.05)	0.817	0.685	0.851	0.623	NDCONV	Div (0.05)	85.3	78.1	95.1	87.2	88.1	79.2	94.8	86.2
DNDCONV	ALWAYS	0.839	0.705	0.845	0.645	DNDCONV	ALWAYS	86.1	78.0	95.3	87.7	87.9	79.0	94.9	87.3
	DEC (150)	0.000	0.650	0.845	0.641	DNDCONV	DEC (150)	65.2	74.7	94.9	87.3	87.6	78.2	94.8	87.0
	Div (0.05)	0.824	0.698	0.848	0.636	DNDCONV	Div (0.05)	84.9	77.2	94.8	87.2	87.7	78.2	94.8	87.1
FLOWNETS	ALWAYS	0.768	0.668	0.774	0.617	FLOWNETS	ALWAYS	83.7	74.1	94.0	84.9	84.7	74.8	93.3	84.5
	DEC (150)	0.031	0.604	0.760	0.559	FLOWNETS	DEC (150)	64.5	72.3	93.8	84.7	84.7	74.6	93.1	84.3
	Div (0.05)	0.771	0.663	0.768	0.612	FLOWNETS	Div (0.05)	83.1	73.7	93.6	84.6	84.4	74.7	93.2	84.4

Table A.9: MOTION-F1 (left) and RECONSTRUCTION ACCURACY (right), comparing neural models and HS algorithm (with different iteration limits and warm start mode). HS algorithm achieves much lower F1 with respect to neural models. The difference is less evident for RECONSTRUCTION ACCURACY.

Model		A	B	Model		A	B	C	M
		HS 30 it (w.s.)	0.574			0.369	HS 30 it (w.s.)	0.830	0.787
HS 30 it	0.649	0.489	HS 30 it	0.879	0.822	0.945	0.867		
HS 200 it (w.s.)	0.462	0.322	HS 200 it (w.s.)	0.827	0.782	0.936	0.846		
HS 200 it	0.683	0.462	HS 200 it	0.816	0.790	0.935	0.886		
LEARNING+	RESUNET	0.858	0.707	LEARNING+	RESUNET	0.902	0.805	0.934	0.880
	NDCONV	0.872	0.693	LEARNING+	NDCONV	0.907	0.790	0.944	0.894
	DNDCONV	0.869	0.715	LEARNING+	DNDCONV	0.895	0.788	0.947	0.878
	FLOWNETS	0.800	0.638	LEARNING+	FLOWNETS	0.863	0.745	0.939	0.850
FROZEN+	RESUNET	0.813	0.664	FROZEN+	RESUNET	0.892	0.817	0.952	0.877
	NDCONV	0.848	0.633	FROZEN+	NDCONV	0.889	0.805	0.952	0.869
	DNDCONV	0.845	0.645	FROZEN+	DNDCONV	0.879	0.790	0.949	0.873
	FLOWNETS	0.774	0.617	FROZEN+	FLOWNETS	0.847	0.748	0.933	0.845

# Appendix B

## Experimental details on attention-based agents (COAT)

This supplemental section includes more details about some aspects of COAT, presented in Sec. 6.4. Code, instructions, and 3D scenes are publicly available<sup>1</sup>.

### B.1 Setup

We created three visual streams ( $256 \times 256$ ), both in grayscale (BW) and color (RGB)—SOLID is BW only—by observing the moving scenes. To accelerate the experimental validation we generated  $\approx 51$  k, 12 k and 20 k pre-rendered frames from the three aforementioned streams, respectively, that correspond to 31 completed laps for each object. The agent learns by watching the first 25 laps per object and, afterwards, he receives 3 supervisions  $(k_t, y_t)$  per object at different time instants, starting from the first frame of the route and waiting at least 100 frames before supervising the same object again (of course,  $a_t$  must belong to the supervised object). When all the objects complete 30 laps, we stopped learning the model parameters and, finally, the last lap was the one on which performance is measured.

**Parameters.** The parameters of the attention model were either fixed ( $\alpha_m = 1$ ), or adapted according to a preliminary run of the model on the video streams ( $\alpha_b, \rho$ ). For each video stream, we searched for the model hyper-parameters that maximize the F1 along the attention trajectory, measured during the 30-th laps, considering the following grids:  $\alpha \in \{5 \cdot 10^{-6}, 10^{-4}, 5 \cdot 10^{-4}, 10^{-3}\}$ ,  $\lambda_T \in \{10^{-2}, 10^{-1}, 1\}$ ,  $\lambda_S \in \{10^{-4}, 10^{-3}, 10^{-2}\}$ ,  $\lambda_C \in \{10^{-4}, 10^{-2}, 10^{-1}, 1, 10\}$ ,  $d \in \{32, 128\}$ ,  $e \in \{5 \text{ k}, 10 \text{ k}, 30 \text{ k}\}$ ,  $\beta \in \{3, 5\}$ .<sup>2</sup>

---

<sup>1</sup>[https://github.com/sailab-code/cl\\_stochastic\\_coherence](https://github.com/sailab-code/cl_stochastic_coherence)

<sup>2</sup>We also evaluated the possibility of removing the outer skip connections in HOURGLASS, and all the models tested normalized feature vectors (unitary norm) and not normalized ones.

## B.2 Normalized representations

In addition to what has been described in Sec. 6.4, we also explored the case in which the feature vectors are normalized in order to have unitary Euclidean norms,

$$\|f_t(x)\| = 1, \quad \forall x \in Z^\diamond, \quad \forall t \geq 0. \quad (\text{B.1})$$

This avoids the system to encode information in the length of the feature vectors, and it can be used to simplify the loss terms of Eq. 6.2, Eq. 6.3, Eq. 6.4 as follows,

$$L_T(f_t, f_{t-1}) := \pi_t (1 - \langle f_t(a_t), f_{t-1}(a_{t-1}) \rangle) \quad (\text{B.2})$$

$$L_S(f_t) := \frac{1}{2} \sum_{\substack{x, z \in S_t \\ x \neq z}} (1 - \langle f_t(x), f_t(z) \rangle) \quad (\text{B.3})$$

$$L_C(f_t) := \sum_{x \in S_t} \sum_{z \in O_t} (1 + \langle f_t(x), f_t(z) \rangle) \quad (\text{B.4})$$

where  $\langle \cdot, \cdot \rangle$  is the dot product and each loss has zero as minimum value. In the case of pixel-wise classification, when using normalized representations we computed the distance between a template  $k_t$  and a certain feature vector  $f_{t'}(x)$  (with  $t' \geq t$ ) as

$$\text{dist}(k_t, f_{t'}(x)) = 1 - \langle k_t, f_{t'}(x) \rangle, \quad (\text{B.5})$$

and the open-set threshold  $\zeta$  belongs to  $(0, 2]$ .

## B.3 Segmenting the attended moving region

The coordinates belonging to set  $S_t$  are those of the pixels that belong to the moving region that includes the attention coordinates  $a_t$ . If  $v_{x,t}$  is the 2D velocity of pixel at coordinates  $x$  and time  $t$ , then such pixel is a *moving pixel* if and only if it is associated to a non-vanishing optical flow, i.e.,

$$\|v_{x,t}\| > \gamma, \quad (\text{B.6})$$

given  $\gamma > 0$  and being  $\|\cdot\|$  the Euclidean norm. Two *moving pixels* that are direct neighbors in the image plane are defined as *connected*, and a chain of connected pixels implements what we will refer to as a *path*.

We introduce the function  $\text{path\_exists}(x, a_t)$ , that returns true if there exists a path connecting  $x$  to the attention  $a_t$ . Hence,

$$S_t = \{x: \text{path\_exists}(x, a_t)\} \cup \{a_t\}. \quad (\text{B.7})$$

If there are no moving pixels in the current frame, then we consider  $S_t$  to be empty (in this case, also  $a_t \notin S_t$ ), and the spatial coherence and contrastive losses are set to zero.



In order to segment the moving area as defined above, we implemented a frontier-based algorithm reported in Alg. 6, where the function  $\text{neighbors}(x)$  returns the set of direct neighbors of  $x$ .

---

**Algorithm 6** Determining the moving area that includes the focus of attention.

---

**Input:** Optical flow  $[v_{x,t}]_{x \in Z^\circ}$ , attention coordinates  $a_t$ , threshold  $\gamma > 0$ .

**Output:** Set of coordinates belonging to the moving area, i.e.,  $S_t$ .

```

1:  $S_t \leftarrow \{a_t\}$  {▷ initial set}
2:  $\mathcal{F} \leftarrow \{a_t\}$  {▷ initial frontier}
3: while  $\mathcal{F} \neq \emptyset$  do
4:    $\mathcal{N} \leftarrow \cup_{x \in \mathcal{F}} \text{neighbors}(x)$  {▷ union of the neighbors of the points in the frontier}
5:    $\mathcal{F} \leftarrow \{z : z \in \mathcal{N} \wedge z \notin S_t \wedge \|v_{z,t}\| > \gamma\}$  {▷ new frontier: moving pixels in  $\mathcal{N}$  (not in  $S_t$ )}
6:    $S_t \leftarrow S_t \cup \mathcal{F}$  {▷ adding the new frontier to  $S_t$ }
7: end while
8: if  $|S_t| = 1$  then
9:    $S_t \leftarrow \emptyset$  {▷ if only  $a_t$  belongs to  $S_t$ , we clear it}
10: end if
11: return  $S_t$ 

```

---

## B.4 Neural architectures

The experimental campaign was carried out evaluating the performance of the proposed approach using two different types of deep convolutional architectures with  $d$  output filters, referred to as RESUNET and FCN-ND, respectively. The former is a UNet (Ronneberger et al., 2015) architecture<sup>3</sup> based on a ResNet18 backbone. Differently, the other model we considered, FCN-ND, is a 6-layer Fully-Convolutional model that maintains the same image-resolution throughout the whole architecture, without any downsamplings/poolings, inspired by Sherrah (2016). It is based on  $5 \times 5$  filters, with 6 layers composed of 32 filters in each hidden layer, except from the first one, which contains 16 filter banks. In the experiments, we report the average results obtained over 3 runs initialized with random seeds. We report in Table B.1 the hyperparameters corresponding to the best models. See the provided code repository for further details (e.g, on the network variations hyperparameters.)

---

<sup>3</sup><https://github.com/usuyama/pytorch-unet> (MIT License).

		EMPTYSPACE		SOLID	LIVINGROOM	
Parameters		BW	RGB	BW	BW	RGB
RESUNET	$\alpha$	$5 \cdot 10^{-6}$	$5 \cdot 10^{-4}$	$10^{-4}$	$10^{-3}$	$5 \cdot 10^{-4}$
	$\lambda_T$	1	1	$10^{-1}$	$10^{-1}$	$10^{-1}$
	$\lambda_S$	$10^{-4}$	$10^{-4}$	$10^{-3}$	$10^{-2}$	$10^{-3}$
	$\lambda_C$	$10^{-1}$	$10^{-1}$	$10^{-1}$	10	10
	$d$	32	128	128	32	32
	$e$	30 k	30 k	10 k	30 k	30 k
	$\beta$	5	5	5	5	5
	NORM.	yes	yes	yes	yes	yes
FCN-ND	$\alpha$	$10^{-3}$	$10^{-3}$	$10^{-3}$	$10^{-4}$	$10^{-3}$
	$\lambda_T$	$10^{-1}$	$10^{-2}$	$10^{-2}$	$10^{-2}$	$10^{-2}$
	$\lambda_S$	$10^{-2}$	$10^{-3}$	$10^{-2}$	$10^{-2}$	$10^{-3}$
	$\lambda_C$	$10^{-2}$	$10^{-2}$	$10^{-2}$	1	$10^{-1}$
	$d$	32	32	32	128	128
	$e$	10 k	10 k	10 k	30 k	30 k
	$\beta$	5	3	3	3	3
	NORM.	no	no	no	yes	yes

Table B.1: Optimal parameters. Best selected hyperparameters drawn from the grid search described in the main text (see Sec. 5.4), both for RESUNET (top) and FCN-ND (bottom) architectures. Additionally, we denote with  $\text{NORM.} \in \{yes, no\}$  the choice of processing normalized feature vectors (unitary norm) and not normalized ones, respectively. For further details, see the provided code repository.

# Appendix C

## Experimental details on motion-based agents (CMOSFET)

This appendix provides further details about CMOSFET, the model proposed in Sec. 6.5. This includes in-depth descriptions of the neural architectures and the selected competitors, the grids of values of the hyper-parameters that we validated in our experimental activity, together with the best selected values. We also provide additional results and qualitative investigations. Code, instructions and data are publicly available<sup>1</sup>.

### C.1 Neural Architectures

We provide additional details on the selected neural architectures and on the competitors.

**Feature extractors** The basic block of the feature branch of CMOSFET, i.e., function  $f$ , is a ResUnet architecture, inspired to the one used for COAT but having a smaller amount of parameters. It is a UNet-like (Ronneberger et al., 2015) architecture<sup>2</sup>, based on a ResNet18 backbone. Among the pretrained competitors, DPT-Hybrid (Ranftl et al., 2021) is a Transformer-based ViT model, trained on the ADE20K dataset (Zhou et al., 2019). All the other considered models adopt ResNet backbones, in particular the ResNet-50 architecture, except for DeepLab v3 (ResNet-101). DeepLab models<sup>3</sup> apply Atrous Spatial Pyramid Pooling (ASPP) on top of the backbone features and they are trained on COCO dataset (Lin et al., 2014). MoCo

---

<sup>1</sup><https://github.com/sailab-code/unsupervised-learning-feature-flow>

<sup>2</sup>Inspired to <https://github.com/usuyama/pytorch-unet> (MIT License), but with one fourth of filters in each layer. Also, the last layer has an output dimensionality of 32 features

<sup>3</sup>Available at <https://pytorch.org/vision/stable/models.html>

models<sup>4</sup> are trained with a contrastive unsupervised objective on ImageNet-1M. MoCo v2 differs from the original version because of an improved training procedure (better augmentation, cosine learning rate schedule, extra MLP in training phase). MoCo v3<sup>5</sup> introduces some tricks to improve the learning stability both for ViT and ResNets backbones. We focus our analysis on the latter backbone. In the case of PixPro<sup>6</sup> models, we use backbone weights obtained from pretraining on ImageNet-1M with their pixel-level contrastive task.

**Motion estimation** CMOSFET neural branch for motion estimation, i.e., function  $\delta$ , inherits its structure and composition from the feature extraction one (ResUnet), except that we cut the skip connections connecting the input to the output layer, since we empirically noticed in preliminary experiments that we could obtain a more sensible estimation. Such a network outputs the vertical and horizontal displacement maps for all the frame pixels.

## C.2 Additional implementation details

In order to complement the already described experimental setup of Sec. 5.4, here we introduce some further details to make our experience reproducible. For completeness we report that, in our implementation, the maximum distance between pairs of pixels in Eq. 6.9 (i.e.,  $\sqrt{h^2 + w^2}$  in the normalization factors) is replaced with the maximum distance between pairs of sampled coordinates.

**Data augmentation** For each processed frame pair, we create a mini-batch composed of the two original frames  $(I_{t-1}, I_t)$ , as well as multiple augmented views/transformations, in order to improve the feature robustness, as common in deep net training procedures. Augmentations are of 3 different types: (A) large random crops (with crop ratio greater than 0.9) followed by upscaling to  $w \times h$ ; (B) random horizontal/vertical flips; (C) random color distortion and Gaussian blur. For the augmentations of type (A) and (C), only one of the frames composing the pair  $(I_{t-1}, I_t)$  is transformed, randomly selected at each  $t$ , whilst the other is kept as it is.

**Parameters and validation** In order to make the comparisons fair, we followed the exact same hyper-parameter validation procedure of COAT (Sec. 6.4). In particular, for each video stream we selected the values of the hyper-parameters which maximize the F1 score measured during the 30-th lap, only considering one pixel per

---

<sup>4</sup>Available at <https://github.com/facebookresearch/moco>

<sup>5</sup>Available at <https://github.com/facebookresearch/moco-v3>

<sup>6</sup>Available at <https://github.com/zdaxie/PixPro>

frame, along the trajectory of a human-like focus of attention mechanism attending the scene. We sampled the best hyper-parameters from the following grids:

$$\begin{aligned} \lambda_m &\in \{1\}, \beta_m \in \{10^{-4}, 10^{-3}, 10^{-2}\}, \beta_f \in \{1\}, \tau_p \in \{0.7, 0.8, 0.9\}, \\ \tau_n &\in \{-0.5, -0.3, 0, 0.3, 0.5\}, \ell \in \{50, 100, 200\}, \tau \in \{0.1, 0.5\}, \\ \tau_m &\in \{0.5, 1.0, 1.5, 2.0\}, \alpha_m \in \{10^{-4}, 5 \cdot 10^{-4}, 10^{-3}, 10^{-2}\}, \\ \alpha_f &\in \{10^{-4}, 5 \cdot 10^{-4}, 10^{-3}, 10^{-2}\}, \xi \in \{0.5, 0.9, 0.99\}. \end{aligned}$$

An additional speed-up of the training process can be achieved by further limiting the summations of Eq. 6.10 (both of them) by keeping only a fraction  $\aleph \in (0, 1]$  of the positive and negative pairs (balanced). In particular, we experimented the case in which we kept those positive (resp. negative) pairs for which representations are the least similar (resp. most similar), i.e., focusing on the ones that mostly contribute to larger values of  $L_{self}$ . In the hyperparameter search, we consider  $\aleph \in \{0.5, 0.7, 1.0\}$ . The motion estimator  $\delta$  is optimized using Adam (Kingma and Ba, 2014), while we empirically found better results by optimizing the features extractor  $f$  with SGD (with the exception of the `EMPTYSPACE` stream, where Adam usually yields better performances). In addition to the typical smoothness regularizer introduced in Eq. 6.8, we also exploited a further regularization modulated by a customizable parameter  $\lambda_r$ , in order to keep the estimated motion magnitude small in the case of uniform backgrounds,

$$\mathcal{R}(\delta_t^\ell) = \lambda_s \cdot (wh)^{-1} \sum_{x \in \mathcal{X}} \|\nabla \delta_t^\ell(x)\|^2 + \lambda_r \cdot (wh)^{-1} \sum_{x \in \mathcal{X}} \|\delta_t^\ell(x)\|^2 \quad (\text{C.1})$$

with  $\lambda_r, \lambda_s \in \{10^{-4}, 5 \times 10^{-4}, 10^{-3}, 5 \times 10^{-3}, 10^{-2}\}$ . With reference to Eq. 6.8, in the implementation we have appropriate coefficients for the three terms:

$$\begin{aligned} L_{conj}^\ell &= \lambda_{cur}^\ell \mathcal{L}_c(\delta_t^\ell, f_{t-1}^\ell, f_t^\ell) \\ &\quad + \lambda_{skip}^\ell \mathcal{L}_c(\delta_t^\ell, f_{t-1}^\ell, f_t^\ell) \\ &\quad + \lambda_{low}^\ell \mathcal{L}_c(\delta_t^\ell, f_{t-1}^{\ell-1}, f_t^{\ell-1}) \\ &\quad + \mathcal{R}(\delta_t^\ell) \end{aligned} \quad (\text{C.2})$$

with the abovementioned coefficients in the range  $\{1, 2\} \times \{10^{-5}, 10^{-4}, 10^{-3}, 10^{-2}, 10^{-1}\}$ . Concerning the learning schedule  $T_{sched}$  (motion estimator warmup), we tried the following values  $\{0, 2, 10\}$  laps, where 0 means no scheduling (the entire model is updated starting on the very first frame). In all our experiments, we reported the average results obtained over 10 runs, initializing random generators with different seeds.

We report in Tab. C.1 the best-found values for the hyperparameters.

Table C.1: Optimal parameters. The best selected hyperparameters drawn from the grids described in the text, for all the datasets. See the code for further details.

Parameters	EMPTYSPACE		SOLID	LIVINGROOM		RAT	HORSE
	BW	RGB	BW	BW	RGB	RGB	RGB
$\aleph$	0.7	1.0	1.0	1.0	1.0	0.7	1.0
$\alpha_f$	$10^{-3}$	$10^{-3}$	$10^{-2}$	$10^{-3}$	$10^{-3}$	$10^{-3}$	$10^{-4}$
$\alpha_m$	$10^{-4}$	$10^{-4}$	$10^{-4}$	$10^{-4}$	$10^{-4}$	$5 \times 10^{-4}$	$10^{-4}$
$\eta$	100	100	100	200	200	200	200
$\lambda_{low}^0$	1	1	1	1	1	1	1
$\lambda_{low}^1$	0.1	1	1	1	1	0.01	0.01
$\lambda_{cur}^0$	$10^{-4}$	$10^{-4}$	$10^{-2}$	$10^{-4}$	$10^{-4}$	$10^{-4}$	$10^{-2}$
$\lambda_{cur}^1$	$10^{-2}$	$10^{-4}$	$10^{-3}$	$10^{-2}$	$10^{-2}$	$10^{-4}$	$10^{-2}$
$\lambda_{skip}^0$	0	0	0	0.2	0.2	$2 \times 10^{-5}$	$2 \times 10^{-3}$
$\lambda_{skip}^1$	0	0	0	0.2	0.2	$2 \times 10^{-5}$	$2 \times 10^{-3}$
$\tau$	0.5	0.1	0.5	0.1	0.1	0.5	0.5
$\tau_m$	1.5	1.5	2	0.7	0.7	1.3	1.3
$\tau_n$	0	0	-0.3	-0.5	-0.5	0.5	0.5
$\tau_p$	0.9	0.7	0.7	0.9	0.9	0.8	0.8
$\xi$	0.99	0.5	0.99	0.99	0.99	0.99	0.99
$\lambda_r$	$10^{-3}$	$5 \times 10^{-3}$	$10^{-3}$	$10^{-2}$	$10^{-2}$	$10^{-2}$	$10^{-2}$
$\lambda_s$	$10^{-4}$	$10^{-4}$	$10^{-4}$	$10^{-4}$	$10^{-4}$	$10^{-3}$	$10^{-4}$
$T_{sched}$	10	0	0	0	0	10	2

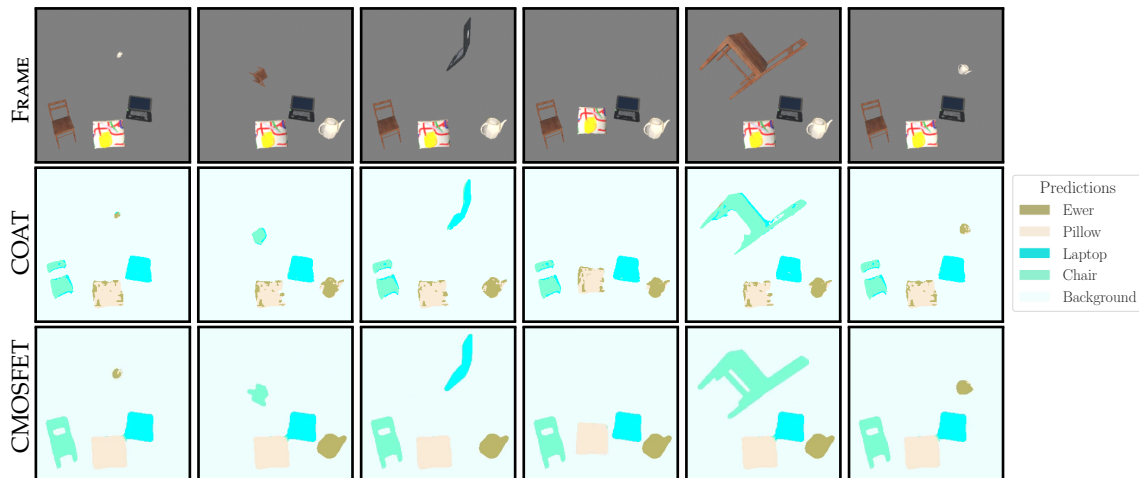


Figure C.1: Six frames sampled from the EMPTYSPACE-RGB stream (first row). We compare the predictions yielded by the COAT (second row) and the outputs by CMOSFET (third row). Different colors are used to denote different classes.

### C.3 Additional qualitative analysis

We report in Fig. C.1 some additional qualitative results of the predictions obtained leveraging the features by CMOSFET, on six frames from the EMPTYSPACE-RGB stream (first row). COAT yields features which are not fully capable to discriminate pixels

from the chair legs/thinner segments, as long as some textures on the pillow: see for instance the borders of the pillow which are misled with the ewer (given the similar brightness/RGB values). Similarly, some parts of the ewer (the spout and the handle) are mistaken for pixels belonging to the pillow. Conversely, CMOSFET allows the classification procedure (which is not involved in the feature learning process) to almost completely disentangle all the different objects and their parts, even when they appear in peculiar poses and orientations (see the chair legs in the fifth image). As a general consideration, CMOSFET yields features (and thus predictions) which tend to slightly overflow with respect to the ground-truth object borders.

# Appendix D

## Publications

### Journal papers

1. **S Marullo**, M Tiezzi, M Gori, S Melacci, “Continual Learning of Conjugated Visual Representations through Higher-order Motion Flows”, *Submitted to TPAMI*. **Candidate’s contributions:** design and implementation of algorithms, experimental campaign, writing of the paper.

### Peer reviewed conference papers

1. M Tiezzi, **S Marullo**, F Becattini, S Melacci, “Continual Memory Neurons”, *Submitted to ICLR*. **Candidate’s contributions:** partial design of algorithms, part of the experimental campaign, partial writing of the paper.
2. A Betti, M Casoni, M Gori, **S Marullo**, S Melacci, M Tiezzi, “Neural Time-Reversed generalized Riccati Equation”, *Submitted to AAAI*. **Candidate’s contributions:** discussions, part of the experimental campaign, review of the paper.
3. **S Marullo**, M Tiezzi, A Betti, M Casoni, S Melacci, “ Bridging Continual Learning of Motion and Self-Supervised Representations”, *Submitted to AAAI*. **Candidate’s contributions:** design and implementation of algorithms, experimental campaign, writing of the paper.
4. **S Marullo**, M Tiezzi, M Gori, S Melacci, T Tuytelaars, “Continual Learning with Pretrained Backbones by Tuning in the Input Space”, *IJCNN*, 2023. **Candidate’s contributions:** design and implementation of algorithms, experimental campaign, writing of the paper.
5. E Meloni, L Faggi, **S Marullo**, A Betti, M Tiezzi, M Gori, S Melacci, “PARTIME: Scalable and Parallel Processing Over Time with Deep Neural Networks”, *IEEE*



- ICMLA*, 2022. **Candidate's contributions:** discussions, partial implementation of the algorithms, part of the experimental campaign, review of the paper.
6. M Tiezzi, **S Marullo**, A Betti, E Meloni, L Faggi, M Gori, S Melacci, "Foveated Neural Computation", *ECML-PKDD*, 2022. **Candidate's contributions:** discussions, part of the experimental campaign, partial writing and review of the paper.
  7. **S Marullo**, M Tiezzi, A Betti, L Faggi, E Meloni, S Melacci, "Continual Unsupervised Learning for Optical Flow Estimation with Deep Networks", *CoLLAs*, 2022. **Candidate's contributions:** design and implementation of algorithms, experimental campaign, writing of the paper.
  8. A Betti, L Faggi, M Gori, M Tiezzi, **S Marullo**, E Meloni, S Melacci, "Continual Learning through Hamilton Equations", *CoLLAs*, 2022. **Candidate's contributions:** discussions, partial implementation of algorithms, review of the paper.
  9. M Tiezzi, **S Marullo**, L Faggi, E Meloni, A Betti, S Melacci, "Stochastic Coherence Over Attention Trajectory For Continuous Learning In Video Streams", *IJCAI*, 2022. **Candidate's contributions:** design and implementation of algorithms, part of the experimental campaign, writing of the paper.
  10. **S Marullo**, M Tiezzi, M Gori, S Melacci, "Being Friends Instead of Adversaries: Deep Networks Learn from Data Simplified by Other Networks", *AAAI*, 2022. **Candidate's contributions:** design and implementation of algorithms, experimental campaign, writing of the paper.
  11. **S Marullo**, M Tiezzi, M Gori, S Melacci, "Friendly Training: Neural Networks Can Adapt Data To Make Learning Easier", *IEEE International Joint Conference on Neural Networks*, 2021. **Candidate's contributions:** design and implementation of algorithms, experimental campaign, writing of the paper.
  12. A Betti, M Gori, **S Marullo**, S Melacci, "Developing Constrained Neural Units Over Time", *IJCNN*, 2020. **Candidate's contributions:** discussions, implementation of algorithms, experimental campaign, partial writing of the paper.

## Workshop papers

1. E Meloni, A Betti, L Faggi, **S Marullo**, M Tiezzi, S Melacci, "Evaluating Continual Learning Algorithms by Generating 3D Virtual Environments", *IJCAI International Workshop on Continual Semi-Supervised Learning*, 2021. **Candidate's contributions:** discussions, partial writing of the paper.

# Bibliography

- Abnar, S., Dehghani, M., Neyshabur, B., and Sedghi, H. (2022). Exploring the limits of large scale pre-training. In *Proc. of the Intl. Conf. on Learning Representations (ICLR)*.
- Achille, A., Rovere, M., and Soatto, S. (2018). Critical learning periods in deep networks. In *International Conference on Learning Representations*.
- Aljundi, R., Chakravarty, P., and Tuytelaars, T. (2017). Expert gate: Lifelong learning with a network of experts. In *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 7120–7129.
- Aljundi, R., Kelchtermans, K., and Tuytelaars, T. (2019). Task-free continual learning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 11254–11263.
- Allgower, E. L. and Georg, K. (2003). *Introduction to Numerical Continuation Methods*. Springer.
- An, S., Li, Y., Lin, Z., Liu, Q., Chen, B., Fu, Q., Chen, W., Zheng, N., and Lou, J.-G. (2022). Input-tuning: Adapting unfamiliar inputs to frozen pretrained models. *arXiv preprint arXiv:2203.03131*.
- Arani, E., Sarfraz, F., and Zonooz, B. (2022). Learning fast, learning slow: A general continual learning method based on complementary learning system. In *International Conference on Learning Representations*.
- Baker, S., Roth, S., Scharstein, D., Black, M. J., Lewis, J., and Szeliski, R. (2007). A database and evaluation methodology for optical flow. In *2007 IEEE 11th International Conference on Computer Vision*, pages 1–8.
- Bengio, Y., Louradour, J., Collobert, R., and Weston, J. (2009). Curriculum learning. In *International conference on machine learning*, page 41–48. PMLR.
- Betti, A., Boccignone, G., Faggi, L., Gori, M., and Melacci, S. (2021a). Visual features and their own optical flow. *Frontiers in Artificial Intelligence*, 4.

- Betti, A., Faggi, L., Gori, M., Tiezzi, M., Marullo, S., Meloni, E., and Melacci, S. (2022a). Continual learning through hamilton equations. In *Conference on Lifelong Learning Agents*, pages 201–212. PMLR.
- Betti, A., Gori, M., and Melacci, S. (2020a). Cognitive action laws: The case of visual features. *IEEE Transactions on Neural Networks and Learning Systems*, 31(3):938–949.
- Betti, A., Gori, M., and Melacci, S. (2020b). Learning visual features under motion invariance. *Neural Networks*, 126:275–299.
- Betti, A., Gori, M., and Melacci, S. (2022b). *Deep Learning to See-Towards New Foundations of Computer Vision*. Springer.
- Betti, A., Gori, M., Melacci, S., Pelillo, M., and Roli, F. (2021b). Can machines learn to see without visual databases? In *NeurIPS Workshop on Data Centric AI - arXiv preprint arXiv:2110.05973*.
- Borji, A. (2019). Saliency prediction in the deep learning era: Successes and limitations. *IEEE transactions on pattern analysis and machine intelligence*.
- Borji, A. and Itti, L. (2012). State-of-the-art in visual attention modeling. *IEEE transactions on pattern analysis and machine intelligence*, 35(1):185–207.
- Braddick, O. and Atkinson, J. (2011). Development of human visual function. *Vision Research*, 51(13):1588–1609. Vision Research 50th Anniversary Issue: Part 2.
- Bregler, C. (1997). Learning and recognizing human dynamics in video sequences. In *Proceedings of IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, pages 568–574.
- Bromley, J., Guyon, I., LeCun, Y., Säcker, E., and Shah, R. (1993). Signature verification using a "siamese" time delay neural network. In Cowan, J., Tesauro, G., and Alspector, J., editors, *Advances in Neural Information Processing Systems*, volume 6. Morgan-Kaufmann.
- Brown, T. B., Mann, B., Ryder, N., Subbiah, M., Kaplan, J., Dhariwal, P., Neelakantan, A., Shyam, P., Sastry, G., Askell, A., et al. (2020). Language models are few-shot learners. In *Advances in Neural Information Processing Systems*.
- Brox, T., Bruhn, A., Papenber, N., and Weickert, J. (2004). High accuracy optical flow estimation based on a theory for warping. In *European Conference on Computer Vision*, pages 25–36. Springer.
- Bulatov, Y. (2011). notmnist dataset. <http://yaroslavvb.blogspot.com/2011/09/notmnist-dataset.html>.

- Burt, R., Thigpen, N. N., Keil, A., and Principe, J. C. (2021). Unsupervised foveal vision neural architecture with top-down attention. *Neural Networks*, 141:145–159.
- Butler, D. J., Wulff, J., Stanley, G. B., and Black, M. J. (2012). A naturalistic open source movie for optical flow evaluation. In A. Fitzgibbon et al. (Eds.), editor, *European Conf. on Computer Vision (ECCV)*, Part IV, LNCS 7577, pages 611–625. Springer-Verlag.
- Buzzega, P., Boschini, M., Porrello, A., Abati, D., and Calderara, S. (2020). Dark experience for general continual learning: a strong, simple baseline. In Larochelle, H., Ranzato, M., Hadsell, R., Balcan, M. F., and Lin, H., editors, *Advances in Neural Information Processing Systems*, volume 33, pages 15920–15930. Curran Associates, Inc.
- Cai, H., Gan, C., Zhu, L., and Han, S. (2020). Tinytl: Reduce memory, not parameters for efficient on-device learning. In *Advances in Neural Information Processing Systems*.
- Carlson, A., Betteridge, J., Kisiel, B., Settles, B., Hruschka, E., and Mitchell, T. (2010). Toward an architecture for never-ending language learning. *Proceedings of the AAAI Conference on Artificial Intelligence*, 24(1):1306–1313.
- Charpiat, G., Hofmann, M., and Schölkopf, B. (2008). Automatic image colorization via multimodal predictions. In Forsyth, D., Torr, P., and Zisserman, A., editors, *Computer Vision – ECCV 2008*, pages 126–139, Berlin, Heidelberg. Springer Berlin Heidelberg.
- Chaudhari, S., Mithal, V., Polatkan, G., and Ramanath, R. (2019). An attentive survey of attention models. *arXiv preprint arXiv:1904.02874*.
- Chaudhry, A., Ranzato, M., Rohrbach, M., and Elhoseiny, M. (2019). Efficient lifelong learning with a-GEM. In *International Conference on Learning Representations*.
- Chen, L.-C., Papandreou, G., Kokkinos, I., Murphy, K., and Yuille, A. L. (2017). Deeplab: Semantic image segmentation with deep convolutional nets, atrous convolution, and fully connected crfs. *IEEE TPAMI*, 40(4):834–848.
- Chen, T., Kornblith, S., Norouzi, M., and Hinton, G. (2020a). A simple framework for contrastive learning of visual representations. In *International conference on machine learning*, pages 1597–1607. PMLR.
- Chen, X., Fan, H., Girshick, R., and He, K. (2020b). Improved baselines with momentum contrastive learning. *arXiv preprint arXiv:2003.04297*.

- Chen, X. and He, K. (2021). Exploring simple siamese representation learning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 15750–15758.
- Chen, X., Xie, S., and He, K. (2021). An empirical study of training self-supervised vision transformers. In *2021 IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 9620–9629, Los Alamitos, CA, USA. IEEE Computer Society.
- Chen, Z. and Liu, B. (2018). *Lifelong machine learning*, volume 1. Springer.
- Cheng, G., Han, J., and Lu, X. (2017). Remote sensing image scene classification: Benchmark and state of the art. *Proceedings of the IEEE*.
- Cloutman, L. L. (2013). Interaction between dorsal and ventral processing streams: Where, when and how? *Brain and Language*, 127(2):251–263.
- Cornia, M., Baraldi, L., Serra, G., and Cucchiara, R. (2016). A deep multi-level network for saliency prediction. In *2016 23rd International Conference on Pattern Recognition (ICPR)*, pages 3488–3493.
- Cossu, A., Tuytelaars, T., Carta, A., Passaro, L., Lomonaco, V., and Bacciu, D. (2022). Continual Pre-Training Mitigates Forgetting in Language and Vision. *arXiv preprint arXiv:2205.09357*.
- De Lange, M., Aljundi, R., Masana, M., Parisot, S., Jia, X., Leonardis, A., Slabaugh, G., and Tuytelaars, T. (2021). A continual learning survey: Defying forgetting in classification tasks. *IEEE transactions on pattern analysis and machine intelligence*, 44(7):3366–3385.
- De Lange, M. and Tuytelaars, T. (2021). Continual prototype evolution: Learning online from non-stationary data streams. In *2021 IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 8230–8239.
- Devlin, J., Chang, M.-W., Lee, K., and Toutanova, K. (2019). BERT: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 4171–4186, Minneapolis, Minnesota. Association for Computational Linguistics.
- Dhar, P., Singh, R. V., Peng, K.-C., Wu, Z., and Chellappa, R. (2019). Learning without memorizing. In *Proc. IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*.
- Dosovitskiy, A., Beyer, L., Kolesnikov, A., Weissenborn, D., Zhai, X., Unterthiner, T., Dehghani, M., Minderer, M., Heigold, G., Gelly, S., Uszkoreit, J., and Houlsby, N.

- (2021). An image is worth 16x16 words: Transformers for image recognition at scale. In *International Conference on Learning Representations*.
- Dosovitskiy, A., Fischer, P., Ilg, E., Häusser, P., Hazirbas, C., Golkov, V., Smagt, P. v. d., Cremers, D., and Brox, T. (2015). FlowNet: Learning optical flow with convolutional networks. In *2015 IEEE International Conference on Computer Vision (ICCV)*, pages 2758–2766.
- Ebrahimi, S., Meier, F., Calandra, R., Darrell, T., and Rohrbach, M. (2020). Adversarial continual learning. *Proc. of European Conference on Computer Vision (ECCV)*.
- Elman, J. L. (1993). Learning and development in neural networks: The importance of starting small. *Cognition*, 48(1):71–99.
- Elsayed, G. F., Goodfellow, I., and Sohl-Dickstein, J. (2019). Adversarial reprogramming of neural networks. In *Proc. of the Intl. Conf. on Learning Representations (ICLR)*.
- Ermis, B., Zappella, G., Wistuba, M., Rawal, A., and Archambeau, C. (2022). Memory efficient continual learning with transformers. In Oh, A. H., Agarwal, A., Belgrave, D., and Cho, K., editors, *Advances in Neural Information Processing Systems*.
- Feldman, J. and Tremoulet, P. D. (2006). Individuation of visual objects over time. *Cognition*, 99(2):131–165.
- Fini, E., da Costa, V. G. T., Alameda-Pineda, X., Ricci, E., Alahari, K., and Mairal, J. (2022). Self-supervised models are continual learners. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 9621–9630.
- French, R. M. (1999). Catastrophic forgetting in connectionist networks. *Trends in Cognitive Sciences*, 3(4):128–135.
- Gan, C., Schwartz, J., Alter, S., Mrowca, D., Schrimpf, M., Traer, J., Freitas, J. D., Kubilius, J., Bhandwadar, A., Haber, N., Sano, M., Kim, K., Wang, E., Lingelbach, M., Curtis, A., Feigelis, K., Bear, D. M., Gutfreund, D., Cox, D., Torralba, A., DiCarlo, J. J., Tenenbaum, J. B., McDermott, J. H., and Yamins, D. L. K. (2021). Threed-world: A platform for interactive multi-modal physical simulation.
- Geirhos, R., Rubisch, P., Michaelis, C., Bethge, M., Wichmann, F. A., and Brendel, W. (2019). Imagenet-trained CNNs are biased towards texture; increasing shape bias improves accuracy and robustness. In *Proc. of the Intl. Conf. on Learning Representations (ICLR)*.

- Geng, C., Huang, S.-j., and Chen, S. (2020). Recent advances in open set recognition: A survey. *IEEE transactions on pattern analysis and machine intelligence*.
- Gepperth, A. and Karaoguz, C. (2016). A bio-inspired incremental learning architecture for applied perceptual problems. *Cognitive Computation*, 8(5):924–934.
- Goodfellow, I., Pouget-Abadie, J., Mirza, M., Xu, B., Warde-Farley, D., Ozair, S., Courville, A., and Bengio, Y. (2020). Generative adversarial networks. *Commun. ACM*, 63(11):139–144.
- Goodfellow, I., Shlens, J., and Szegedy, C. (2015). Explaining and harnessing adversarial examples. In *International Conference on Learning Representations*.
- Gori, M. and Melacci, S. (2023). Collectionless artificial intelligence. *arXiv:2309.06938*.
- GrandViewResearch (2022). Artificial Intelligence Market Size, Share, Growth Report 2030 — grandviewresearch.com. <https://www.grandviewresearch.com/industry-analysis/artificial-intelligence-ai-market>. [Accessed 13-10-2023].
- Grossberg, S. T. (2012). *Studies of mind and brain: Neural principles of learning, perception, development, cognition, and motor control*, volume 70. Springer Science & Business Media.
- Gutmann, M. and Hyvärinen, A. (2010). Noise-contrastive estimation: A new estimation principle for unnormalized statistical models. In Teh, Y. W. and Titterton, M., editors, *Proceedings of the Thirteenth International Conference on Artificial Intelligence and Statistics*, volume 9 of *Proceedings of Machine Learning Research*, pages 297–304, Chia Laguna Resort, Sardinia, Italy. PMLR.
- Hayes, T. L. and Kanan, C. (2020). Lifelong machine learning with deep streaming linear discriminant analysis. In *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*, pages 887–896.
- He, K., Fan, H., Wu, Y., Xie, S., and Girshick, R. (2020). Momentum contrast for unsupervised visual representation learning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 9729–9738.
- He, K., Zhang, X., Ren, S., and Sun, J. (2016). Deep residual learning for image recognition. In *IEEE Conf. on Computer Vision and Pattern Recognition*, pages 770–778.
- Hinton, G., Vinyals, O., and Dean, J. (2014). Distilling the knowledge in a neural network. In *Advances in Neural Information Processing Systems - Workshop on Deep Learning*.

- Hoi, S. C., Sahoo, D., Lu, J., and Zhao, P. (2018). Online learning: A comprehensive survey. *arXiv preprint arXiv:1802.02871*.
- Horn, B. and Schunck, B. (1981). Determining optical flow. *Artificial Intelligence*, 17:185–203.
- Hu, D., Yan, S., Lu, Q., Hong, L., Hu, H., Zhang, Y., Li, Z., Wang, X., and Feng, J. (2022). How well does self-supervised pre-training perform with streaming data? In *Intl. Conf. on Learning Representations (ICLR)*.
- Ioffe, S. and Szegedy, C. (2015). Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *International conference on machine learning*, pages 448–456. PMLR.
- Itti, L., Koch, C., and Niebur, E. (1998). A model of saliency-based visual attention for rapid scene analysis. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 20(11):1254–1259.
- Jaderberg, M., Simonyan, K., Zisserman, A., and Kavukcuoglu, K. (2015). Spatial transformer networks. In *Advances in Neural Information Processing Systems*, volume 28.
- Jain, S., Salman, H., Khaddaj, A., Wong, E., Park, S. M., and Mađdry, A. (2023). A data-based perspective on transfer learning. In *2023 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 3613–3622.
- Jaiswal, A., Babu, A. R., Zadeh, M. Z., Banerjee, D., and Makedon, F. (2021). A survey on contrastive self-supervised learning. *Technologies*, 9(1).
- Jastrzebski, S., Szymczak, M., Fort, S., Arpit, D., Tabor, J., Cho, K., and Geras, K. (2019). The break-even point on optimization trajectories of deep neural networks. In *International Conference on Learning Representations*.
- Jia, M., Tang, L., Chen, B.-C., Cardie, C., Belongie, S., Hariharan, B., and Lim, S.-N. (2022). Visual prompt tuning. In *Proc. of the European Conf. on Computer Vision (ECCV)*.
- Jones, K. S. (1972). A statistical interpretation of term specificity and its application in retrieval. *Journal of Documentation*, 28:11–21.
- Jonschkowski, R., Stone, A., Barron, J. T., Gordon, A., Konolige, K., and Angelova, A. (2020). What matters in unsupervised optical flow. In *European Conference on Computer Vision*, pages 557–572. Springer.
- Kingma, D. P. and Ba, J. (2014). Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*.



- Kingma, D. P. and Ba, J. (2015). Adam: A method for stochastic optimization. In *International Conference on Learning Representations*.
- Kingma, D. P. and Welling, M. (2013). Auto-encoding variational bayes. *arXiv:1312.6114*.
- Kirkpatrick, J., Pascanu, R., Rabinowitz, N., Veness, J., Desjardins, G., Rusu, A. A., Milan, K., Quan, J., Ramalho, T., Grabska-Barwinska, A., et al. (2017). Overcoming catastrophic forgetting in neural networks. *Proc. of the National Academy of Sciences*.
- Kornblith, S., Shlens, J., and Le, Q. V. (2019). Do better imagenet models transfer better? In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*.
- Kowler, E. (2009). *Attention and Eye Movements*, pages 605–616. Elsevier Ltd.
- Krizhevsky, A. (2009). Learning Multiple Layers of Features from Tiny Images. <https://www.cs.toronto.edu/~kriz/learning-features-2009-TR.pdf>.
- Krizhevsky, A., Sutskever, I., and Hinton, G. E. (2012). Imagenet classification with deep convolutional neural networks. *Advances in neural information processing systems*, 25:1097–1105.
- Kumar, M. P., Packer, B., and Koller, D. (2010). Self-paced learning for latent variable models. In *Proceedings of the 23rd International Conference on Neural Information Processing Systems-Volume 1*, pages 1189–1197.
- Larochelle, H., Erhan, D., Courville, A., Bergstra, J., and Bengio, Y. (2007). An empirical evaluation of deep architectures on problems with many factors of variation. In *International conference on machine learning*, pages 473–480. PMLR.
- Lässig, F., Aceituno, P. V., Sorbaro, M., and Grewe, B. F. (2023). Bio-inspired, task-free continual learning through activity regularization. *Biological Cybernetics*, 117(4):345–361.
- LeCun, Y. (1998). The mnist database of handwritten digits. <http://yann.lecun.com/exdb/mnist/>.
- Lecun, Y., Bottou, L., Bengio, Y., and Haffner, P. (1998). Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324.
- Lesort, T., George, T., and Rish, I. (2021). Continual learning in deep networks: an analysis of the last layer. *arXiv preprint arXiv:2106.01834*.

- Lester, B., Al-Rfou, R., and Constant, N. (2021). The power of scale for parameter-efficient prompt tuning. In *Proc. of the 2021 Conf. on Empirical Methods in Natural Language Processing*. Assoc. for Computational Linguistics.
- Li, H. and Gong, M. (2017). Self-paced convolutional neural networks. In *Proceedings of the 26th International Joint Conference on Artificial Intelligence*, pages 2110–2116.
- Li, X. L. and Liang, P. (2021). Prefix-tuning: Optimizing continuous prompts for generation. *Proc. of the 59th Annual Meeting of the Assoc. for Computational Linguistics and the 11th Intl. Joint Conf. on Natural Language Processing*, abs/2101.00190.
- Li, Z. and Hoiem, D. (2017). Learning without forgetting. *IEEE Transactions on Pattern Analysis and Machine Intelligence*.
- Liang, Y., Li, X., Jafari, N., and Chen, J. (2020). Video object segmentation with adaptive feature bank and uncertain-region refinement. In Larochelle, H., Ranzato, M., Hadsell, R., Balcan, M., and Lin, H., editors, *Advances in Neural Information Processing Systems*, volume 33, pages 3430–3441. Curran Associates, Inc.
- Lin, T.-Y., Maire, M., Belongie, S., Hays, J., Perona, P., Ramanan, D., Dollár, P., and Zitnick, C. L. (2014). Microsoft coco: Common objects in context. In *ECCV*, pages 740–755. Springer.
- Lindsay, G. W. (2021). Convolutional Neural Networks as a Model of the Visual System: Past, Present, and Future. *Journal of Cognitive Neuroscience*, 33(10):2017–2031.
- Lomonaco, V. and Maltoni, D. (2017). Core50: a new dataset and benchmark for continuous object recognition. In Levine, S., Vanhoucke, V., and Goldberg, K., editors, *Proceedings of the 1st Annual Conference on Robot Learning*, volume 78 of *Proceedings of Machine Learning Research*, pages 17–26. PMLR.
- Long, J., Shelhamer, E., and Darrell, T. (2015). Fully convolutional networks for semantic segmentation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 3431–3440.
- Lopez-Paz, D. and Ranzato, M. (2017). Gradient episodic memory for continual learning. In *Proceedings of the 31st International Conference on Neural Information Processing Systems, NIPS’17*, page 6470–6479, Red Hook, NY, USA. Curran Associates Inc.
- Lucas, B. D. and Kanade, T. (1981). An iterative image registration technique with an application to stereo vision. In *Proceedings of the 7th International Joint Conference on Artificial Intelligence - Volume 2, IJCAI’81*, page 674–679, San Francisco, CA, USA. Morgan Kaufmann Publishers Inc.

- Maas, A. L., Daly, R. E., Pham, P. T., Huang, D., Ng, A. Y., and Potts, C. (2011). Learning word vectors for sentiment analysis. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies*, pages 142–150, Portland, Oregon, USA. Association for Computational Linguistics.
- Madaan, D., Yoon, J., Li, Y., Liu, Y., and Hwang, S. J. (2022). Representational continuity for unsupervised continual learning. In *International Conference on Learning Representations*.
- Madry, A., Makelov, A., Schmidt, L., Tsipras, D., and Vladu, A. (2018). Towards deep learning models resistant to adversarial attacks. In *International Conference on Learning Representations*.
- Mahendran, A., Thewlis, J., and Vedaldi, A. (2019). Cross pixel optical-flow similarity for self-supervised learning. In *Computer Vision—ACCV 2018: 14th Asian Conference on Computer Vision, Perth, Australia, December 2–6, 2018, Revised Selected Papers, Part V 14*, pages 99–116. Springer.
- Mahendran, A. and Vedaldi, A. (2015). Understanding deep image representations by inverting them. In *Proc. of the IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*.
- Mai, Z., Li, R., Jeong, J., Quispe, D., Kim, H., and Sanner, S. (2022). Online continual learning in image classification: An empirical survey. *Neurocomputing*, 469:28–51.
- Marullo, S., Tiezzi, M., Betti, A., Faggi, L., Meloni, E., and Melacci, S. (2022a). Continual unsupervised learning for optical flow estimation with deep networks. In *Conference on Lifelong Learning Agents*, pages 183–200. PMLR.
- Marullo, S., Tiezzi, M., Gori, M., and Melacci, S. (2021). Friendly training: Neural networks can adapt data to make learning easier. In *Proc. of the Intl. Joint Conf. on Neural Networks (IJCNN)*.
- Marullo, S., Tiezzi, M., Gori, M., and Melacci, S. (2022b). Being friends instead of adversaries: Deep networks learn from data simplified by other networks. In *Proc. of the AAAI Conf. on Artificial Intelligence*.
- Marullo, S., Tiezzi, M., Gori, M., and Melacci, S. (2023a). Continual learning of conjugated visual representations through higher-order motion flows. Unpublished.
- Marullo, S., Tiezzi, M., Gori, M., Melacci, S., and Tuytelaars, T. (2023b). Continual learning with pretrained backbones by tuning in the input space. In *2023 International Joint Conference on Neural Networks (IJCNN)*, pages 1–9.

- Masana, M., Liu, X., Twardowski, B., Menta, M., Bagdanov, A. D., and van de Weijer, J. (2022). Class-incremental learning: survey and performance evaluation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*.
- McCloskey, M. and Cohen, N. J. (1989). Catastrophic interference in connectionist networks: The sequential learning problem. In *Psychology of learning and motivation*, volume 24, pages 109–165. Elsevier.
- Mehta, S. V., Patil, D., Chandar, S., and Strubell, E. (2023). An empirical investigation of the role of pre-training in lifelong learning. *Journal of Machine Learning Research*, 24(214):1–50.
- Meloni, E., Pasqualini, L., Tiezzi, M., Gori, M., and Melacci, S. (2021). Sailenv: Learning in virtual visual environments made simple. In *2020 25th International Conference on Pattern Recognition (ICPR)*, pages 8906–8913. IEEE.
- Milner, A. D. (2017). How do the two visual streams interact with each other? *Experimental brain research*, 235(5):1297–1308.
- Min, J., Kang, D., and Cho, M. (2021). Hypercorrelation squeeze for few-shot segmentation. *arXiv preprint arXiv:2104.01538*.
- Misra, I. and van der Maaten, L. (2020). Self-supervised learning of pretext-invariant representations. In *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 6706–6716.
- Mitchell, T., Cohen, W., Hruschka, E., Talukdar, P., Betteridge, J., Carlson, A., Dalvi Mishra, B., Gardner, M., Kisiel, B., Krishnamurthy, J., Lao, N., Mazaitis, K., Mohamed, T., Nakashole, N., Platanios, E., Ritter, A., Samadi, M., Settles, B., Wang, R., Wijaya, D., Gupta, A., Chen, X., Saparov, A., Greaves, M., and Welling, J. (2015). Never-ending learning. *Proceedings of the AAAI Conference on Artificial Intelligence*, 29(1).
- Mitchell, T. M. (1997). *Machine Learning*. McGraw-Hill, Inc., New York.
- Morerio, P., Cavazza, J., Volpi, R., Vidal, R., and Murino, V. (2017). Curriculum dropout. In *2017 IEEE International Conference on Computer Vision (ICCV)*, pages 3564–3572, Los Alamitos, CA, USA. IEEE Computer Society.
- Murshed, M. G. S., Murphy, C., Hou, D., Khan, N., Ananthanarayanan, G., and Husain, F. (2021). Machine learning at the network edge: A survey. *ACM Comput. Surv.*
- Netzer, Y., Wang, T., Coates, A., Bissacco, A., Wu, B., and Ng, A. Y. (2011). Reading digits in natural images with unsupervised feature learning. In *Advances in Neural Information Processing Systems*.

- NextMSC (2023). Ai market: Global opportunity analysis and industry forecast — grandviewresearch.com. <https://www.nextmsc.com/report/artificial-intelligence-market>. [Accessed 13-10-2023].
- Noroozi, M. and Favaro, P. (2016). Unsupervised learning of visual representations by solving jigsaw puzzles. In Leibe, B., Matas, J., Sebe, N., and Welling, M., editors, *Computer Vision – ECCV 2016*, pages 69–84, Cham. Springer International Publishing.
- Oord, A. v. d., Li, Y., and Vinyals, O. (2018). Representation learning with contrastive predictive coding. *arXiv preprint arXiv:1807.03748*.
- Ostapenko, O., Lesort, T., Rodriguez, P., Arefin, M. R., Douillard, A., Rish, I., and Charlin, L. (2022). Continual Learning with Foundation Models: An Empirical Study of Latent Replay. In *Proc. of the 1st Conf. on Lifelong Learning Agents (CoLLAs)*. PMLR.
- Ostrovsky, Y., Meyers, E., Ganesh, S., Mathur, U., and Sinha, P. (2009). Visual parsing after recovery from blindness. *Psychological Science*, 20(12):1484–1491.
- Ozbulak, U., Lee, H. J., Boga, B., Anzaku, E. T., min Park, H., Messem, A. V., Neve, W. D., and Vankerschaver, J. (2023). Know your self-supervised learning: A survey on image-based generative and discriminative training. *Transactions on Machine Learning Research*. Survey Certification.
- Palazzi, A., Abati, D., Calderara, s., Solera, F., and Cucchiara, R. (2019). Predicting the driver’s focus of attention: The dr(eye)ve project. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 41(7):1720–1733.
- Papert, S. A. (1966). The Summer Vision Project — dspace.mit.edu. <https://dspace.mit.edu/handle/1721.1/6125?show=full>. [Accessed 15-10-2023].
- Parisi, G. I., Kemker, R., Part, J. L., Kanan, C., and Wermter, S. (2019). Continual lifelong learning with neural networks: A review. *Neural networks*, 113:54–71.
- Pathak, D., Girshick, R., Dollár, P., Darrell, T., and Hariharan, B. (2017). Learning features by watching objects move. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 2701–2710.
- Pathak, D., Krähenbühl, P., Donahue, J., Darrell, T., and Efros, A. A. (2016). Context encoders: Feature learning by inpainting. In *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 2536–2544.
- Pellegrini, L., Graffieti, G., Lomonaco, V., and Maltoni, D. (2020). Latent replay for real-time continual learning. In *Proc. of the IEEE Intl. Conf. on Intelligent Robots and Systems (IROS)*.

- Peng, X., Bai, Q., Xia, X., Huang, Z., Saenko, K., and Wang, B. (2019). Moment matching for multi-source domain adaptation. In *Proc. of the IEEE Intl. Conf. on Computer Vision (ICCV)*.
- Peterson, G. (2004). A day of great illumination: B. F. Skinner’s discovery of shaping. *Journal of the experimental analysis of behavior*, 82:317–28.
- Pfeiffer, J., Rücklé, A., Poth, C., Kamath, A., Vulić, I., Ruder, S., Cho, K., and Gurevych, I. (2020). Adapterhub: A framework for adapting transformers. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, pages 46–54.
- Pham, Q., Liu, C., and HOI, S. (2021). Dualnet: Continual learning, fast and slow. In Beygelzimer, A., Dauphin, Y., Liang, P., and Vaughan, J. W., editors, *Advances in Neural Information Processing Systems*.
- Phuong, M. and Lampert, C. (2019). Towards understanding knowledge distillation. In *International Conference on Machine Learning*, volume 97, pages 5142–5151. PMLR.
- Qian, R., Meng, T., Gong, B., Yang, M.-H., Wang, H., Belongie, S., and Cui, Y. (2021). Spatiotemporal contrastive video representation learning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 6964–6974.
- Qiu, H., Xiao, C., Yang, L., Yan, X., Lee, H., and Li, B. (2020). Semanticadv: Generating adversarial examples via attribute-conditioned image editing. In *ECCV*.
- Raghunathan, A., Xie, S. M., Yang, F., Duchi, J. C., and Liang, P. (2019). Adversarial training can hurt generalization. In *International Conference on Machine Learning 2019 - Workshop Deep Phenomena*.
- Ramasesh, V. V., Dyer, E., and Raghu, M. (2021). Anatomy of catastrophic forgetting: Hidden representations and task semantics. In *Proc. of the Intl. Conf. on Learning Representations (ICLR)*.
- Ramasesh, V. V., Lewkowycz, A., and Dyer, E. (2022). Effect of scale on catastrophic forgetting in neural networks. In *Proc. of the Intl. Conf. on Learning Representations (ICLR)*.
- Ranftl, R., Bochkovski, A., and Koltun, V. (2021). Vision transformers for dense prediction. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 12179–12188.

- Rao, D., Visin, F., Rusu, A., Pascanu, R., Teh, Y. W., and Hadsell, R. (2019). Continual unsupervised representation learning. In Wallach, H., Larochelle, H., Beygelzimer, A., d'Alché-Buc, F., Fox, E., and Garnett, R., editors, *Advances in Neural Information Processing Systems*, volume 32. Curran Associates, Inc.
- Rebuffi, S.-A., Kolesnikov, A., Sperl, G., and Lampert, C. H. (2017). icarl: Incremental classifier and representation learning. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*.
- Reimers, N. and Gurevych, I. (2019). Sentence-bert: Sentence embeddings using siamese bert-networks. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing*. ACL.
- Rolnick, D., Ahuja, A., Schwarz, J., Lillicrap, T., and Wayne, G. (2019). Experience replay for continual learning. In Wallach, H., Larochelle, H., Beygelzimer, A., d'Alché-Buc, F., Fox, E., and Garnett, R., editors, *Advances in Neural Information Processing Systems*, volume 32. Curran Associates, Inc.
- Ronneberger, O., Fischer, P., and Brox, T. (2015). U-net: Convolutional networks for biomedical image segmentation. In Navab, N., Hornegger, J., Wells, W. M., and Frangi, A. F., editors, *Medical Image Computing and Computer-Assisted Intervention (MICCAI)*, pages 234–241. Springer.
- Rucci, M., McGraw, P. V., and Krauzlis, R. J. (2016). Fixational eye movements and perception. *Vision Research*, 118:1–4. Fixational eye movements and perception.
- Rucci, M. and Poletti, M. (2015). Control and functions of fixational eye movements. *Annual Review of Vision Science*, 1:499–518.
- Russakovsky, O., Deng, J., Su, H., Krause, J., Satheesh, S., Ma, S., Huang, Z., Karpathy, A., Khosla, A., Bernstein, M., Berg, A. C., and Fei-Fei, L. (2015). ImageNet Large Scale Visual Recognition Challenge. *Intl. Journal of Computer Vision (IJCV)*.
- Russell, S. and Norvig, P. (2009). *Artificial Intelligence: A Modern Approach*. Prentice Hall Press, USA, 3rd edition.
- Rusu, A. A., Rabinowitz, N. C., Desjardins, G., Soyer, H., Kirkpatrick, J., Kavukcuoglu, K., Pascanu, R., and Hadsell, R. (2022). Progressive neural networks. *arXiv preprint arXiv:1606.04671*.
- Scheirer, W. J., de Rezende Rocha, A., Sapkota, A., and Boulton, T. E. (2012). Toward open set recognition. *IEEE transactions on pattern analysis and machine intelligence*, 35(7):1757–1772.

- Sherrah, J. (2016). Fully convolutional networks for dense semantic labelling of high-resolution aerial imagery. *arXiv preprint arXiv:1606.02585*.
- Sinha, S., Garg, A., and Larochelle, H. (2020). Curriculum by smoothing. In Larochelle, H., Ranzato, M., Hadsell, R., Balcan, M., and Lin, H., editors, *Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems 2020, NeurIPS 2020, December 6-12, 2020, virtual*.
- Smith, L., Yu, C., Yoshida, H., and Fausey, C. (2014). Contributions of head-mounted cameras to studying the visual environments of infants and young children. *Journal of Cognition and Development*, 16:141217110914002.
- Smith, L. B. and Slone, L. K. (2017). A developmental approach to machine learning? *Frontiers in Psychology*, 8.
- Spelke, E. S. (1990). Principles of object perception. *Cognitive science*, 14(1):29–56.
- Srivastava, N., Hinton, G., Krizhevsky, A., Sutskever, I., and Salakhutdinov, R. (2014). Dropout: A simple way to prevent neural networks from overfitting. *Journal of Machine Learning Research*, 15(56):1929–1958.
- Stone, A., Maurer, D., Ayvaci, A., Angelova, A., and Jonschkowski, R. (2021). Smurf: Self-teaching multi-frame unsupervised raft with full-image warping. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 3887–3896.
- Sun, D., Roth, S., and Black, M. J. (2010). Secrets of optical flow estimation and their principles. In *International Conference on Computer Vision and Pattern Recognition*, pages 2432–2439.
- Tamis-LeMonda, C. S. and Masek, L. R. (2023). Embodied and embedded learning: Child, caregiver, and context. *Current Directions in Psychological Science*, 32(5):369–378.
- Tan, C., Sun, F., Kong, T., Zhang, W., Yang, C., and Liu, C. (2018). A survey on deep transfer learning. In Kůrková, V., Manolopoulos, Y., Hammer, B., Iliadis, L., and Maglogiannis, I., editors, *Artificial Neural Networks and Machine Learning – ICANN 2018*, pages 270–279, Cham. Springer International Publishing.
- Tarvainen, A. and Valpola, H. (2017). Mean teachers are better role models: Weight-averaged consistency targets improve semi-supervised deep learning results. In Guyon, I., Luxburg, U. V., Bengio, S., Wallach, H., Fergus, R., Vishwanathan, S., and Garnett, R., editors, *Advances in Neural Information Processing Systems*, volume 30. Curran Associates, Inc.



- Teed, Z. and Deng, J. (2020). Raft: Recurrent all-pairs field transforms for optical flow.
- Thoutt, Z. (2017). Wine Reviews: <https://kaggle.com/zynicide/wine-reviews>.
- Thrun, S. and Mitchell, T. M. (1995). Lifelong robot learning. *Robotics and Autonomous Systems*, 15(1):25–46. The Biology and Technology of Intelligent Autonomous Agents.
- Tiezzi, M., Marullo, S., Becattini, F., and Melacci, S. (2023a). Continual memory neurons. In *Submitted to The Twelfth International Conference on Learning Representations*. under review.
- Tiezzi, M., Marullo, S., Betti, A., Meloni, E., Faggi, L., Gori, M., and Melacci, S. (2023b). Foveated neural computation. In *Machine Learning and Knowledge Discovery in Databases: European Conference, ECML PKDD 2022, Grenoble, France, September 19–23, 2022, Proceedings, Part III*, page 19–35, Berlin, Heidelberg. Springer-Verlag.
- Tiezzi, M., Marullo, S., Faggi, L., Meloni, E., Betti, A., and Melacci, S. (2022). Stochastic coherence over attention trajectory for continuous learning in video streams. In *Proceedings of the Thirty-First International Joint Conference on Artificial Intelligence, IJCAI-22*, pages 3480–3486.
- Tiezzi, M., Melacci, S., Betti, A., Maggini, M., and Gori, M. (2020). Focus of attention improves information transfer in visual features. In Larochelle, H., Ranzato, M., Hadsell, R., Balcan, M. F., and Lin, H., editors, *Advances in Neural Information Processing Systems*, volume 33, pages 22194–22204. Curran Associates, Inc.
- Tolstikhin, I. O., Houlsby, N., Kolesnikov, A., Beyer, L., Zhai, X., Unterthiner, T., Yung, J., Steiner, A., Keysers, D., Uszkoreit, J., Lucic, M., and Dosovitskiy, A. (2021). Mlp-mixer: An all-mlp architecture for vision. In Ranzato, M., Beygelzimer, A., Dauphin, Y., Liang, P., and Vaughan, J. W., editors, *Advances in Neural Information Processing Systems*, volume 34, pages 24261–24272. Curran Associates, Inc.
- Tsipras, D., Santurkar, S., Engstrom, L., Turner, A., and Madry, A. (2019). Robustness may be at odds with accuracy. *arXiv preprint arXiv:1805.12152*.
- van de Ven, G. M., Tuytelaars, T., and Tolias, A. S. (2022). Three types of incremental learning. *Nature Machine Intelligence*, 4(12):1185–1197.
- Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, L. u., and Polosukhin, I. (2017). Attention is all you need. In *Advances in Neural Information Processing Systems*.

- Viola, P. and Jones, M. (2001). Rapid object detection using a boosted cascade of simple features. In *Proceedings of the 2001 IEEE Computer Society Conference on Computer Vision and Pattern Recognition. CVPR 2001*.
- Wang, L., Zhang, X., Su, H., and Zhu, J. (2023). A comprehensive survey of continual learning: Theory, method and application. *arXiv preprint arXiv:2302.00487*.
- Wang, X., Zhang, R., Shen, C., Kong, T., and Li, L. (2021). Dense contrastive learning for self-supervised visual pre-training. In *2021 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 3023–3032.
- Wang, Z., Zhang, Z., Ebrahimi, S., Sun, R., Zhang, H., Lee, C.-Y., Ren, X., Su, G., Perot, V., Dy, J., and Pfister, T. (2022a). Dualprompt: Complementary prompting for rehearsal-free continual learning. In *ECCV 2022*.
- Wang, Z., Zhang, Z., Lee, C.-Y., Zhang, H., Sun, R., Ren, X., Su, G., Perot, V., Dy, J., and Pfister, T. (2022b). Learning to prompt for continual learning. In *Proc. of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*.
- Wertheimer, M. (1938). Laws of organization in perceptual forms. *A source book of Gestalt psychology*, pages 71–88.
- Wu, X., Dyer, E., and Neyshabur, B. (2021). When do curricula work? In *International Conference on Learning Representations*.
- Wu, Z., Xiong, Y., Yu, S. X., and Lin, D. (2018). Unsupervised feature learning via non-parametric instance discrimination. In *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 3733–3742.
- Xiao, C., Li, B., yan Zhu, J., He, W., Liu, M., and Song, D. (2018). Generating adversarial examples with adversarial networks. In *Proceedings of the Twenty-Seventh International Joint Conference on Artificial Intelligence, IJCAI-18*, pages 3905–3911. International Joint Conferences on Artificial Intelligence Organization.
- Xiao, H., Rasul, K., and Vollgraf, R. (2017). Fashion-mnist: a novel image dataset for benchmarking machine learning algorithms. *arXiv preprint arXiv:1708.07747*.
- Xie, Z., Lin, Y., Zhang, Z., Cao, Y., Lin, S., and Hu, H. (2021). Propagate yourself: Exploring pixel-level consistency for unsupervised visual representation learning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 16684–16693.
- Xiong, Y., Ren, M., Zeng, W., and Waabi, R. (2021). Self-supervised representation learning from flow equivariance. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 10171–10180.

- Ye, F. and Bors, A. G. (2022). Task-free continual learning via online discrepancy distance learning. In Oh, A. H., Agarwal, A., Belgrave, D., and Cho, K., editors, *Advances in Neural Information Processing Systems*.
- Zanca, D. and Gori, M. (2017). Variational laws of visual attention for dynamic scenes. In *Advances in Neural Information Processing Systems*, pages 3823–3832.
- Zanca, D., Melacci, S., and Gori, M. (2020). Gravitational laws of focus of attention. *IEEE TPAMI*, 42(12):2983–2995.
- Zenke, F., Poole, B., and Ganguli, S. (2017). Continual learning through synaptic intelligence. In *Proc. of the Intl. Conf. on Machine Learning (ICML)*.
- Zeno, C., Golan, I., Hoffer, E., and Soudry, D. (2021). Task-Agnostic Continual Learning Using Online Variational Bayes With Fixed-Point Updates. *Neural Computation*.
- Zhai, M., Xiang, X., Lv, N., and Kong, X. (2021). Optical flow and scene flow estimation: A survey. *Pattern Recognition*, 114:107861.
- Zhang, C., Zhang, K., Zhang, C., Pham, T. X., Yoo, C. D., and Kweon, I. S. (2021). How does SimSiam avoid collapse without negative samples? A unified understanding with self-supervised contrastive learning. In *International Conference on Learning Representations*.
- Zhang, J., Xu, X., Han, B., Niu, G., Cui, L., Sugiyama, M., and Kankanhalli, M. (2020a). Attacks which do not kill training make adversarial learning stronger. In *International conference on machine learning*. PMLR.
- Zhang, Z., Lathuiliere, S., Ricci, E., Sebe, N., Yan, Y., and Yang, J. (2020b). Online depth learning against forgetting in monocular videos. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*.
- Zhou, B., Zhao, H., Puig, X., Xiao, T., Fidler, S., Barriuso, A., and Torralba, A. (2019). Semantic understanding of scenes through the ade20k dataset. *IJCV*, 127(3):302–321.