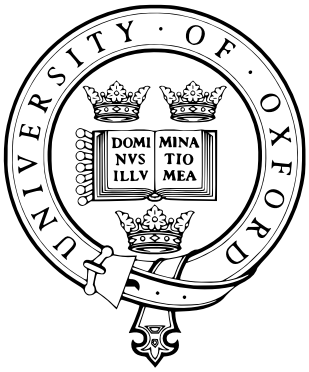
— LEARNING (TO LEARN) FROM FEW EXAMPLES —

VISUAL TRACKING AND CLASSIFICATION WITH LIMITED DATA

LUCA BERTINETTO



Department of Engineering Science
University of Oxford, St. Catherine's College

Hilary Term, 2019

A dissertation submitted in partial fulfillment of
the requirements for the degree of Doctor of Philosophy

ADVISOR
Professor Philip H.S. Torr

Dedicated to my dear parents and sister.
Thank you for sending your love and support from far away.

ABSTRACT

The availability of large labelled datasets has played a crucial role in the recent success of deep neural networks. However, there are many situations in which training data is scarce. For instance, in the case of object tracking, such a limitation arises directly from the definition of the problem, which requires an estimate of the position of an object of interest in every frame of a video with the sole supervision of a bounding box in the first frame. Another scenario which can be doomed by data scarcity is classification. For example, one might want to identify the species of a rare insect with a single reference image, or train a tumour classification system from a few labelled MRI scans. For both tracking and classification, we propose techniques that sidestep per-task data scarcity by leveraging a large number of small episodes, each characterised by a limited training set. This strategy is particularly novel for tracking, for which for many years the standard approach has been to train a discriminative model exclusively *online*, while the video is streaming. Moreover, we examine how this general approach can be framed as *learning to learn*, in the sense that the knowledge distilled *within* a training task is accrued and used *across* tasks. We show how such a framework allows one to devise systems that can be trained with a small amount of per-task data while also being dynamically tailored to the problem at hand. From a practical point of view, the proposed methods have a common focus on simplicity, efficiency and speed, achieved by exploiting the shape or redundancies of the data.

All the code produced in this thesis is open source and available at `www.robots.ox.ac.uk/~luca`.

Keywords : *machine learning, computer vision, visual object tracking, few-shot learning, deep learning, meta-learning, metric learning, similarity learning, Siamese network, Correlation Filter, ridge regression.*

PUBLICATIONS

The following publications contain preliminary reports of the findings in this thesis. Equal contributions are marked with a $*$.

L. Bertinetto, J. F. Henriques, P. H. S. Torr, and A. Vedaldi. Meta-learning with differentiable closed-form solvers. In *International Conference on Learning Representations*, 2019.

L. Bertinetto, J. Valmadre, S. Golodetz, O. Miksik, and P. H. S. Torr. Staple: Complementary learners for real-time tracking. In *IEEE Conference on Computer Vision and Pattern Recognition*, 2016.

L. Bertinetto*, J. Valmadre*, J. F. Henriques, A. Vedaldi, and P. H. S. Torr. Fully-convolutional siamese networks for object tracking. In *European Conference on Computer Vision workshops*, 2016.

J. Valmadre*, L. Bertinetto*, J. F. Henriques, A. Vedaldi, and P. H. S. Torr. End-to-end representation learning for correlation filter based tracking. In *IEEE Conference on Computer Vision and Pattern Recognition*, 2017.

Publications in related topics were also made during the course of this thesis.

L. Bertinetto, J. Valmadre, S. Golodetz, O. Miksik, and P. H. S. Torr. The importance of estimating object extent when tracking with correlation filters. *Technical report*, 2015.

L. Bertinetto*, J. F. Henriques*, J. Valmadre*, P. H. S. Torr, and A. Vedaldi. Learning feed-forward one-shot learners. In *Advances in Neural Information Processing Systems*, 2016.

J. Valmadre*, L. Bertinetto*, J. F. Henriques, R. Tao, A. Vedaldi, A. Smeulders, P. H. S. Torr, and E. Gavves*. Long-term tracking in the wild: A benchmark. In *European Conference on Computer Vision*, 2018.

Q. Wang*, L. Zhang*, L. Bertinetto*, W. Hu, and P. H. S. Torr. Fast online object tracking and segmentation: A unifying approach. In *IEEE Conference on Computer Vision and Pattern Recognition*, 2019.

M. Kristan, A. Leonardis, J. Matas, M. Felsberg, R. Pflugfelder, L. Čehovin Zajc, T. Vojír, G. Häger, A. Lukežič, A. Eldesokey, G. Fernandez, , et al. The Visual Object Tracking VOT challenge results. In *ICCV and ECCV workshops*, 2015, 2016, 2017, 2018.

ACKNOWLEDGMENTS

I would like to use this space to thank everyone who has contributed to making the years of my DPhil fantastic and unforgettable. It is not an empty ritual – I am truly glad to look back and realise that people have played such an important role.

First, I would like to thank Prof. Philip Torr for having believed in me when I first applied and then for having guided me during the most challenging moments of my doctorate. Then, a huge 'thank you!' goes to Jack Valmadre and João Henriques: for having been such great friends and for having taught me so much – you have been a fundamental point of reference. I also thank Andrea Vedaldi, Efstratios Gavves, Stuart Golodetz, Ondrej Miksik, Ran Tao, Qiang Wang and Li Zhang for their great collaboration, and Andrew Zisserman and Jiri Matas for their precious feedbacks.

To everyone at TVG and at FiveAI-Oxford: sharing the lab with you has been a great pleasure. I will never forget our adventurous pre- and post-conference trips and the many fascinating lunch conversations.

To all my friends in Oxford and especially Pedro, Alex, Yossi, Liana, Mikesh, Jana, Beth, Nathan, Corinne, Carla, Julian, Roxanna: you have been like a family to me and you made me feel loved and grounded.

To Niamh, my love – thank you for having been so supportive, sweet and patient. We met in a particularly stressful moment for me and yet you still managed to see something behind that bundle of nerves.

Finally, I would like to thank my family. To my sister Serena: as throughout my whole life, during these years you have been a great and wise friend I could always rely on, despite the distance. To Elena and Marco: you have been truly extraordinary parents. You have shown so much support in all my decisions and always encouraged me to pursue my ambitions and feed my curiosity. I will never be grateful enough for this.

CONTENTS

LIST OF FIGURES

LIST OF TABLES

STATEMENT OF AUTHORSHIP

The original papers from which the four central chapters of this thesis stem are multi-authored, as it is common in scientific subjects. In the following, we specify the extent of the contribution of each author.

We roughly divide the effort going into a paper into four components. **Brainstorming**: conception and progress discussion; **coding**: implementation of the core method; **experiments**: data preparation, implementation of ablative studies, execution of the experiments and collection of the results; **writing**: preparation of the manuscript for submission to peer review.

- Chapter 1. Entirely authored by Luca Bertinetto.
- Chapter 2. Entirely authored by Luca Bertinetto.
- Chapter 3.

  Luca Bertinetto: brainstorming, coding, experiments, writing.

  João Henriques: brainstorming, writing.
- Chapter 4.

  Luca Bertinetto: brainstorming, coding, experiments, writing.

  Jack Valmadre: brainstorming, writing.
- Chapter 5.

  Luca Bertinetto: brainstorming, coding, experiments, writing.

  Jack Valmadre (joint first author): brainstorming, coding, writing.
- Chapter 6.

  Luca Bertinetto: brainstorming, coding, experiments, writing.

  Jack Valmadre (joint first author): brainstorming, coding, writing. Correlation Filter derivatives of Section 6.4.
- Chapter 7. Entirely authored by Luca Bertinetto.

When omitted from the above list, the authors contributed to the brainstorming and reviewed the manuscript.

1

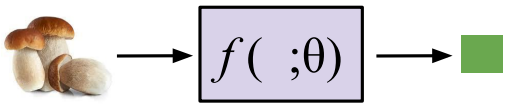
INTRODUCTION

1.1 SUPERVISED LEARNING

The term *supervised learning* is used to identify the family of machine learning algorithms able to leverage labelled training data (the "supervisory signal") to establish a relationship between input data and output values [186]. In recent years, supervised learning algorithms have led to spectacular results in a variety of tasks. As a consequence, they are now used by millions of people every day. Not only have they become essential tools in several academic disciplines such as computer vision [172], high-energy physics [12], genetics [130] and astronomy [13], but they also power countless applications: from trading on financial markets to precision medicine. As a rule of thumb, if *a)* a problem can be formulated as a prediction task and *b)* a significant corpus of labelled data is available, then supervised learning can be useful to tackle this problem.

The general principle of supervised learning remains the same among the growing number of areas in which it is being adopted. During the so-called phase of *training*, by "observing" common patterns emerging from multiple examples, it is possible to establish associations between input data and meaningful properties of the underlying entities that these data portray [22, Chapter 1]. The hope is that, if there are features that are particularly predictive, these will be automatically exploited as clues. For instance, let us imagine we

(a) Training



(b) Inference

Figure 1.1: We can see *training* as a process that receives a training set of many tuples <sample,label> as input and outputs a prediction function $f()$ parametrised by θ, which is able (during *inference*) to associate new samples to their label. Different labels are represented by different colors.

are interested in learning a function to predict whether a fruit is ripe or not. Intuitively, its colour is a more predictive characteristic than its shape or size.

Although, these guidelines are not explicitly embedded in the design of the supervised learning algorithm. Instead, the desired function is "trained" by adapting its internal free parameters, so that a certain error is minimised. This concept is at the core of iterative methods such as Stochastic Gradient Descent (SGD) [26], which can be used to train deep neural networks (DNNs) [76, Chapter 6]. After training, during *inference*, it is crucial for the function to be able to generalise to unseen data. This means that, when exposed to new examples (that have not been shown during training), the function should be able to preserve a reasonable predictive capability [22, Chapter 1].

Since supervised learning algorithms are de facto "association machines" trying to map representations of the real world (data) to certain values we care about, a paramount requirement for their deployment is an adequate amount of labelled data. For instance, Open Images V4 [119] contains more than 9 mil-

lion images with labels for several computer vision tasks: image classification, object detection and visual relationship detection. YouTube-8M [2] and Kinetics [103] boast respectively eight million video sequences labelled with almost 5,000 visual entities and 300,000 short clips covering 400 human action classes. Common Voice [1] is an ongoing project by the Mozilla foundation that has already collected short recordings from tens of thousands of people around the world, amounting to more than 1,600 hours of annotated speech. The wealth of large labelled datasets has undoubtedly pushed forward the boundaries of what is achievable by the field. Importantly, it has enabled DNNs to make good use of their large capacity and deliver impressive results in many disciplines [125].

1.2 LEARNING WITH LIMITED DATA

Although many large and diverse labelled datasets exist, there are situations in which only a limited amount of "samples per concept" is available, thus making the problem of recognising common patterns particularly hard. A frequent scenario is the one in which data is abundant, but labels are highly costly to acquire, a problem that arises when working for instance with autonomous vehicles. Collecting street data is cheap, but high-precision labelling of every video frame is cumbersome, to the point that synthetic datasets [67, 180, 231] are becoming an appealing alternative to real data. Similarly, in drug discovery, project budgets often severely limit the amount of biological data that can be used to form predictions about the properties of compounds of interest [4].

In other circumstances, the problem definition itself could put constraints on the amount of labelled data provided as the supervisory signal. This is the case of arbitrary object tracking [33], in which the only supervision that

identifies the target object is provided *online* and at the first frame, while the video is streaming.

When only a limited amount of data is available, training a function able to generalise becomes much harder. In fact, rather than finding meaningful patterns related to the task, a function's expressive capacity can end up being used to model certain traits of the data that have little or no causal relationship with the desired output. This phenomenon is commonly called *overfitting* [29]. For example, let us imagine that the only dataset available for our ripeness prediction function solely consists of two labelled samples: a photo of a ripe apple and one of a green banana. It is likely that irrelevant information for this problem, such as the fruit's shape, will be interpreted as relevant by the learnt function, so that it will fail to generalise to unseen data (e.g. a ripe banana).

With such limiting circumstances, a conservative strategy [76, Chapters 5 and 7] would be to heavily rely on prior knowledge (for example under the form of regularisation) and to adopt models with little capacity to avoid overfitting. Besides their large capacity and data hunger, in these circumstances DNNs are a bad choice because they require to be trained with gradient-based optimisation techniques such as SGD, which are typically slow. This becomes impractical in a scenario where it is useful to solve many "small" learning problems quickly.

The problem-specific lack of training data can be sidestepped by training a function with a large number of related "episodes". In the few-shot learning literature, for example, an episode is a subset of a larger dataset and it is characterised by a small number of samples of unseen classes, thus mimicking the data-limited scenario that is encountered at inference time [17, 174, 222]. This is the general principle behind many modern *meta-learning* methods.

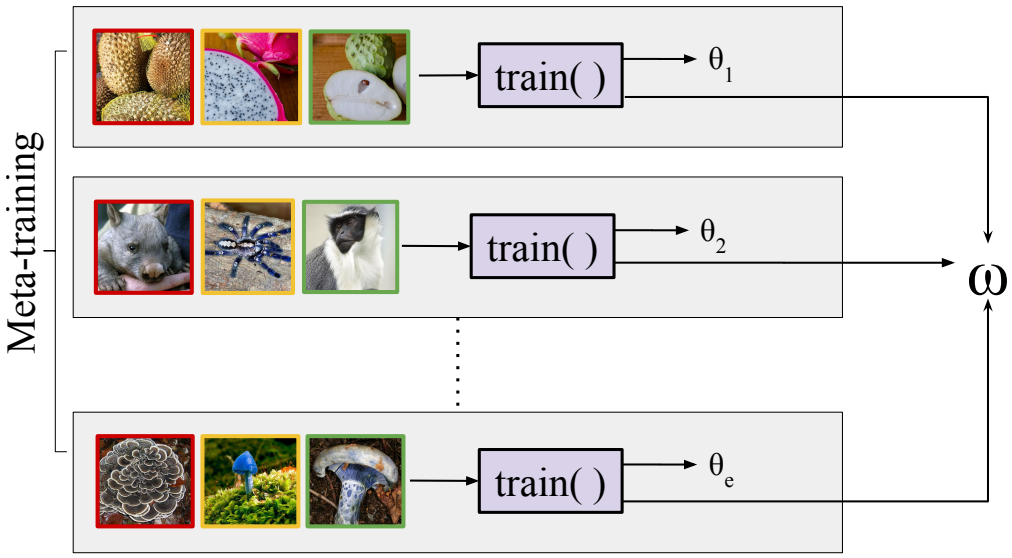


Figure 1.2: Meta-training spans several training *episodes*, each characterised by a small number of samples representing unseen classes. Inside each episode e, a new set of parameters θ_e (corresponding to a new function $f_e()$ with local scope) is trained. At the same time, the training procedure also affects the set of parameters ω, which have global scope and can correspond, for example, to the weights of a DNN. In this way, the lack of training data can be overcome by creating many episodes and allowing knowledge to be accrued over episodes [208, 220]. Different (episode-specific) labels are represented by different colors.

1.3 META-LEARNING

Albeit challenging, it is possible for a learning system to acquire a novel concept despite only little data about that concept is available. It is well known that infants can learn a new word after only a single exposure, a phenomenon called *fast mapping* [31]. How is such a thing possible?

Learning is not restricted to occur within isolated episodes: to acquire new concepts, humans use the knowledge primitives accumulated in a lifetime and adapt them to new evidence [169]. Moreover, with the *overhypothesis* theory, Goodman [50, 77] postulated that humans are able to go beyond the specificity of few examples to derive general principles about categories and properties. These theories of learning are now finding biological analogies thanks to important strides made by neuroscience. For example, Mack et al. [138]

discovered that existing conceptual knowledge residing in the hippocampi is compared by the pre-frontal cortex against relevant features of new data in order to update knowledge. Hence, within our biological learning system, learning new things does not happen in isolation. Quite the opposite: throughout our existence, we are continually updating and refining what we already know. Even though systemic analogies between machine learning algorithms and the human brain are still in their infancy, it is reasonable to draw a connection between the two from a purely epistemic point of view. Similarly to humans, machine learning systems can learn from one or few examples, as long as this process does not happen in a vacuum.

The term *meta-learning* (i.e. *learning to learn*) is often used to describe this type of learning framework, in which the knowledge [142] distilled within a learning episode accrues the general knowledge acquired across episodes [208, 220]. In this way, the problem of (episode-specific) labelled data scarcity can be sidestepped by sampling a large number of episodes, each containing a small training dataset. Figures 1.1 and 1.2 offer a schematic representation of this concept, illustrating the differences between traditional training and meta-training.

This learning framework is very general and can be employed in many different scenarios. However, as previously explained, maintaining a general knowledge that can transcend individual episodes becomes particularly important when the amount of training data within an episode is limited because the straightforward application of dedicated models would result in high overfitting. Therefore, it has become fairly standard to evaluate meta-learning algorithms on *few-shot classification* benchmarks such as Omniglot [123] or *mini*ImageNet [222]. Unlike standard classification, in which often the number of examples per class is in the order of thousands [184], for few-shot classification this number is very small, typically one or five.

Figure 1.3: *Arbitrary object tracking.* Given the supervision of a rectangular bounding box identifying an arbitrary object in the first frame of a video, the problem is to estimate the object's position in all subsequent frames. In many applications of video understanding, tracking is a fundamental component as it allows one to obtain semantic coherence between frames.

In this thesis, besides few-shot classification, we also deal with another computer vision problem characterised with limited amount of training data: arbitrary object tracking. Given the position and the spatial extent of an object of interest in the first frame of a video sequence, the problem consists in estimating the object's location in all the following frames with the best possible accuracy (Figure 1.3). Tracking is a paramount component of many video systems: it enables applications such as surveillance, augmented reality, video editing, sports analytics, drone navigation and autonomous driving [33]. Moreover, it can also be used in a weakly-supervised scenario to create tracks between sparse annotations to increase the number of available labels [229].

In the past, most of the tracking community's efforts [108, 109, 110, 198, 233, 239] have been directed towards learning a representation of the target object exclusively online, while the video is streaming (Figure 1.4). This strategy has been undoubtedly successful, as the accuracy of tracking systems improved significantly over the past years [113, 233]. Compared to few-shot

classification, the conditions under which a learning system for object tracking operates are less extreme, as it is possible to augment the limited training set provided in the first frame by using the estimates of the object's position in new frames as additional samples. Albeit common, this approach inherently creates a spurious model by introducing imperfect predictions into the learning process, which have the effect of amplifying prediction errors [9]. However, perhaps because of the lack of large labelled video datasets, so far the dominant paradigm adopted by real-time trackers has been to confine the process of training exclusively to the "online" phase. Moreover, the low-data regime that characterises this problem setup and its affinities with few-shot classification and meta-learning have been relatively neglected, confining the use of "prior knowledge" to the exploitation of efficient sampling strategies (e.g. [25, 88]) and carefully engineered features for object representation (e.g. [43, 60]).

1.5 CONTRIBUTIONS AND THESIS OUTLINE

This thesis presents several novel methods that tackle the problems of arbitrary object tracking and few-shot classification. Each of the four central chapters (from 3 to 6) corresponds to a standalone published paper that underwent a peer-review process and, as such, reflects a different moment of the author's doctorate. With the intention of making this work more cohesive, the text of the original papers has been partially edited.

Even though multiple themes can be identified throughout this work, one of the most important is the limited data regime common to both computer vision problems taken into account.

An outline of each chapter's content follows.

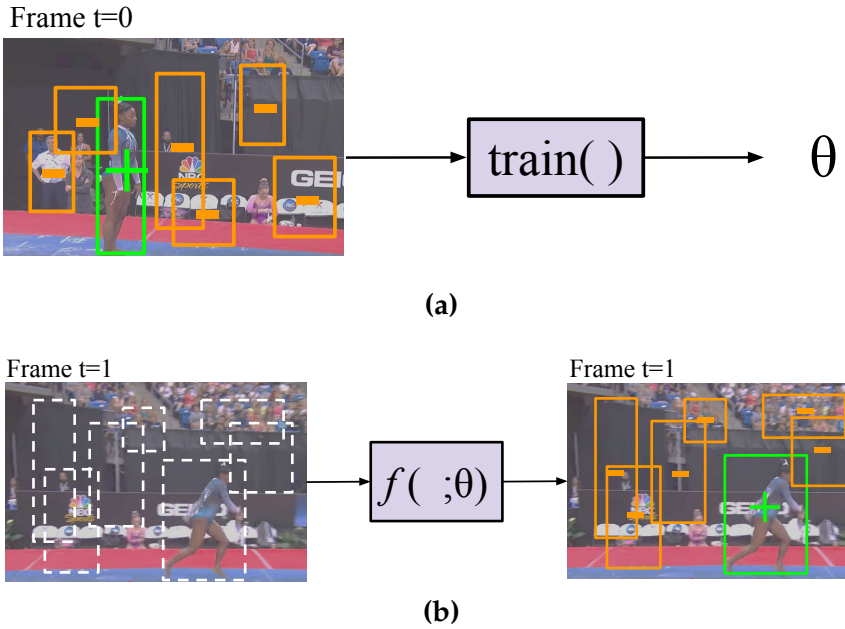


Figure 1.4: Traditional tracking methods train a model of the object of interest (here represented by a function f with parameters θ) with data sampled from the first frame of the video. The learnt function is then evaluated (and potentially updated) in the following frames to estimate new positions of the target. Notice how, while negative samples are abundant, only a single positive ground-truth label is available. With this limited amount of data, it is perhaps overly optimistic to expect to be able to tackle significant transformations that can occur later in a video. To compensate for the scarcity of positive training data, it is common practice to update the model with data sampled from new frames, using the tracker's estimates as additional labels. However, this strategy may come at the significant cost of introducing prediction errors into the learning process [9].

- Before presenting the main contributions, in Chapter 2 we review some of the most relevant prior art. The purpose is to give a general overview of the main literature related to arbitrary object tracking (Section 2.1) and few-shot classification (Section 2.2). Note that the main chapters contain separate *related work* sections, more specific to their very content. Moreover, Section 2.3 briefly outlines the history of meta-learning algorithms, from the evolutionary approach of Schmidhuber [191] to the current state of the art.
- In Chapter 3 [20], we first describe a general framework that can encompass several meta-learning techniques. Then, we introduce a novel approach that learns to learn by backpropagating through simple and

fast converging learners. The main idea is to teach a DNN to use standard machine learning tools, such as logistic regression, as part of its internal model, enabling it to adapt to novel data quickly. This requires backpropagating errors through the solver steps. While usually the cost of the matrix operations involved in such a process would be significant, by using the Woodbury identity [167] we can make the small number of examples work to our advantage. We propose both closed-form and iterative solvers, based on ridge regression and logistic regression components. Our methods constitute a simple and novel approach to the problem of few-shot classification and achieve performance competitive with or superior to the state of the art on three benchmarks.

- Chapter 4 [18] introduces *Staple*, a method for object tracking that adopts the standard strategy of constraining the process of training within the video provided at test time. It shows how a baseline method that simply combines efficient online learners trained with complementary features can perform favourably against the state of the art while also operating at high framerates. This chapter also offers the space to discuss the limitations imposed by learning exclusively from data available at test-time, motivating one of the core contributions of this thesis, illustrated in Chapter 5.
- Chapter 5 [19] introduces a novel tracking framework that significantly differs from the prior art. In this work, instead of being treated as an online learning problem, tracking is performed as the iterative evaluation of a similarity function trained offline on a large video dataset. Importantly, the fully-convolutional aspect of the Siamese network proposed in this chapter allows for efficient sampling and real-time evaluation. Despite its simplicity and the lack of online adaptation, this strategy proves to be both effective and fast.

- Although learning a similarity metric exclusively offline is an effective technique for tracking, it is still desirable to make good use of the annotations provided with the test sequence and update the model online. Therefore, in Chapter 6 [218] we extend the fully-convolutional Siamese framework with meta-learning, showing how to backpropagate through the Correlation Filter (CF) and illustrating when this is advantageous. The CF is an algorithm that trains a linear template to discriminate between images and their translations. It is well suited to object tracking because its formulation in the Fourier domain provides a fast solution. Previous works, however, have adopted features that were either manually designed or trained for a different task. Our work is the first to overcome this limitation by interpreting the Correlation Filter learner as a differentiable layer in a DNN. Moreover, we show that our method has the significant practical benefit of allowing lightweight architectures to achieve a strong performance at high framerates.
- Finally, in Chapter 7 we conclude by outlining the main findings of the thesis and by suggesting potentially interesting avenues for future research.

2

LITERATURE REVIEW

2.1 ARBITRARY OBJECT TRACKING

We are interested in the problem of *online arbitrary object tracking*, which requires a method (the *tracker*) to determine the location of an object of interest provided in the first frame of a video [198]. Most benchmarks (e.g. [58, 97, 219, 232]) adopt an axis-aligned rectangular bounding box to represent the target, while more accurate strategies like rotated bounding boxes [113] and pixel-wise binary masks [170, 228] are less common. *Online* means that the tracker is "causal" [113] and thus it does not make use of future frames as training data, as instead it is common in the video object segmentation community [165]. *Arbitrary* implies that no prior knowledge regarding the target object is made available and thus a tracker should be able to deal with *any* object.

Moreover, to better focus on the problem of dealing with complex appearance changes, arbitrary object tracking benchmarks typically require to track a single object at the time. Instead, multiple object tracking (MOT) benchmarks (e.g. [143]) often present situations in which all the targets belong to the same category and their locations are provided as ground-truth information. In this way, researchers can focus on the problem of establishing the correct data associations between frames, i.e. connecting the locations into identity-aware tracks.

For the sake of simplicity, popular tracking benchmarks have been designed for *short-term* trackers, in the sense that they only contain sequences in which the target object is always visible in the scene. As a consequence, with few exceptions (e.g. [80, 102, 137, 230]), most of the tracking methods published in the past have not been able, by design, to deal with major occlusion or disappearance of the target object. Interestingly, this trend could change in the near future: among the novel tracking benchmarks that have been proposed recently, three explicitly address the long-term scenario [134, 148, 219].

The problem of online arbitrary object tracking presents several important challenges, which we outline below.

- The fact that any arbitrary object of interest might be labelled as the target implies that tracking methods are required to be *class agnostic*. This means that prior knowledge with class-specific information cannot be leveraged directly.
- Moreover, since the object of interest is specified online, trackers can only rely on the ground-truth information provided in the first frame of a video. This is a minimal amount of supervision if we consider that moving objects can undergo complex transformations such as the ones illustrated in Figure 2.1.
- What is more, the small amount of supervision provided often takes the form of an axis-aligned rectangular bounding box [58, 97, 151, 219, 233], which can be inadequate to represent the target object since the very first frame.
- Finally, since arbitrary object trackers are supposed to be used online, an implicit desideratum is that the operating speed should exceed typical video frame-rates.

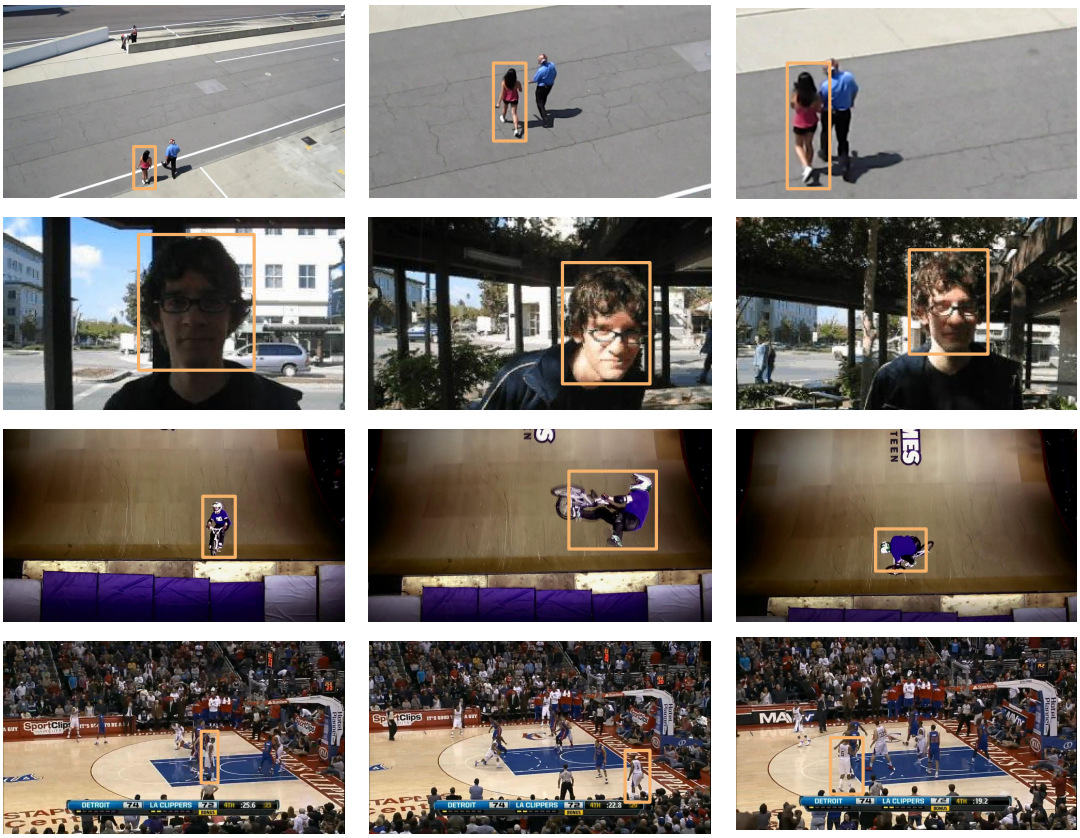


Figure 2.1: Example frames from sequences belonging to OTB-2015 [233] and VOT-2016 [112], two of the most commonly used benchmarks for arbitrary object tracking. Sequences can exhibit challenging transformations of the target object such as scale change (row 1), illumination change (row 2), severe deformation (row 3) and confusion (when several objects look similar to the target, row 4).

Past and present

With the purpose of contextualising the contributions of this thesis, we now describe some of the most salient methods proposed in the past and how the literature evolved towards the current state of the art, with a particular focus on methods based on the Correlation filter and deep neural networks. For a more thorough review of the existing tracking literature, the reader is invited to consult Yilmaz et al. [239], Chen [35] and Smeulders et al. [198]. For the most recent state of the art, instead, the yearly VOT challenge reports [108, 109, 110, 112, 114, 115] are more indicated.

Initially conceived for image registration, the Kanade-Lucas-Tomasi (KLT) algorithm [11, 132] is one of the first and most widely adopted methods for arbitrary object tracking. It uses gradient descent to find the parameters of the transformation which maximises the similarity between the appearance of the target object between frames. Although its initial formulation was limited to simple translations, its extensions are able to tackle affine transformations [65, 141] and illumination changes [82].

Approaches like KLT are effective only if the frame-to-frame transformations can be described by specific parametrised mappings (e.g. affine). Though this strategy goes a long way, it is inherently limited to a constrained set of transformations and types of object. For example, human bodies perform complex non-rigid transformations, making KLT-like approaches ineffective. A way to address this issue is to avoid trying to explicitly parametrise the transformation taking place and instead merely treat tracking as a template matching problem. That is, tracking is solved by looking for the patch in the current frame which is mostly similar to a template representing the target object. Arguably, Normalized Cross-Correlation [27] is one of the simplest examples of this paradigm. Another popular algorithm is the one proposed by Comaniciu et al. [39]. In this case, the target object is represented by an RBG histogram and its location in new frames is determined by finding the mode of the Bhattacharyya distance function via Mean Shift [66]. In general, since colour histograms are statistical representations which discard spatial information, they are particularly robust to substantial deformations of the target object and have been used extensively for object tracking (e.g. [18, 21, 166, 171]). To avoid failure in circumstances such as occlusion or pose change, Adam et al. [3] proposed to subdivide the target object into *fragments*, each represented by a histogram and contributing to the estimate of the overall object's position and scale via a voting mechanism.

Another popular paradigm is to train, online and in the first frame of a video, a discriminative classifier able to distinguish between the object of interest and everything else (the background). Then, the classifier can be updated online, using the estimate of the latest position as new "ground truth" to obtain more training data. In the past, learning algorithms such as online boosting [162] (e.g. [8, 9, 78]), support vector machines (e.g. [7, 83]) and random forests (e.g. [102, 187]) have all been used within this general framework.

Ross et al. [182] have been among the first to recognise the importance of carefully updating the appearance model of the target online, as this can go through a wide variety of extrinsic and intrinsic transformations. They chose an eigenbasis as appearance model but, unlike the prior work of Black and Jepson [23], they proposed to train it online and incrementally as the video progresses. Since recent frames are more appropriate to model the latest target appearance, Ross et al. progressively down-weight older samples by incorporating a forgetting factor in the eigenbasis update. Babenko et al. [9] observed that, when updating the classifier at every frame, small errors in the estimations of the target position can quickly propagate and contaminate the training process. They proposed to resort to multiple-instance learning (MIL) [51], which relies on weakly-supervised sets of samples, where a label is only assigned to a set, not to the individual sample. In this way, the damage caused by a single example inaccurately labelled by the tracker can be contained.

Discriminative classifiers have been (and still are) a popular choice among the tracking community. However, they can become impractical as their computational cost grows with the number of candidate positions, which can be very large if multiple scales and aspect ratios are considered. Instead of learning discriminative classifiers, an alternative strategy is to use regression-based methods, which instead learn a direct relationship between the target's observations and its state (e.g. in the form of a displacement). To learn this corre-

spondence, Williams et al. [230] recur to a Bayesian treatment of the SVM – the Relevance Vector Machine (RVM) [210] – to produce probabilistic regression outputs representing the displacement of the target, which can be expressed by an arbitrary number of degrees of freedom. Rather than learning a single complex model, Zimmermann et al. [248] instead learn an optimal sequence of simple regressors, each initialised with the output of the previous one in a coarse-to-fine approach.

The Correlation Filter. A crucial development in the field of arbitrary object tracking has been the adoption of the Correlation Filter [118] (CF), an algorithm able to discriminate between a template and its translations. The CF can be formulated in the Fourier domain, thus replacing costly correlations with element-wise multiplications. This makes it particularly efficient and allows to densely sample new candidates around the target's previous position. Contrarily, most tracking-by-detection methods are constrained to random sampling for computational reasons. Though its first appearances date back to the eighties [92, 140], it is just with the seminal work of Bolme et al. [25] from 2010 that this algorithm started to be used by the tracking community. Bolme et al. showed that the CF can be used in a simple tracking algorithm to achieve state-of-the-art performance, while running at hundreds of frames per second. Understandably, this paper inspired a great deal of progress in the field. Still today, many of the newly published tracking methods have the CF at their core. Henriques et al. [87, 88] proposed an alternative formulation by treating the sampling space as a circulant matrix. In particular, using HOG features [60], their multi-channel version achieved strong tracking accuracy at high speed. Despite its efficacy, the CF assumes a training set composed by only a single "real" positive example and many "synthetic" negative examples, constituted by circular shifts of the positive example. This assumption causes the so-called "boundary effect", which hinders its performance.

To tackle this problem, Kiani Galoogahi et al. [105] proposed to modify the CF objective with a masking matrix that identifies which parts of the signal should be active and which should not. With the same motivation, Danelljan et al. [45] suggested to use a regularisation component to penalise the filter coefficients proportionally to their distance from the target. Both [105] and [45] arbitrarily mask or penalise locations according to their distance from the image patch centre, while the tracker of Lukežič et al. [133] maintains a binary mask of the target object to better spatially constrain the filter.

Deep learning. After 2012 [117], most areas of computer vision have been profoundly transformed by deep neural networks (DNNs) [125]. However, adoption has been significantly slower in the tracking community, where still in 2015 most methods were based on "shallow" features such as HOG [110] and colour names [43]. A possible reason behind this phenomenon could be the lack, at the time, of large video datasets with per-frame annotations. Arguably, one of the first tracking algorithms to make use of a neural network has been proposed by Wang and Yeung [225]. First, a stacked denoising autoencoder [221] is pre-trained offline on natural images to obtain a robust feature representation. Online, a sigmoid classification layer is added to the encoder, and state estimation is carried out via the particle filter framework [54].

More recent and better performing methods exploit pre-trained DNNs (e.g. [44, 156, 200, 224, 227]) and often adapt them with SGD during tracking [156, 200, 224, 227]. These methods pushed forward the boundaries of the field and achieved unprecedented performance. However, progress came at the cost of speed. State-of-the-art methods such as MDNet [156] and VITAL [200], for example, run at one frame per second or less, making real-time video applications unfeasible. To underline the importance of tracking speed, in 2017 the VOT committee has introduced a "real-time" sub-challenge [114] in their popular yearly workshop.

In our work, illustrated in Chapter 5 and originally presented in [19], we propose an alternative paradigm that allows circumventing this problem by training, offline and end-to-end, a DNN representing a similarity function. Once trained, the network can be simply evaluated at high speed during tracking. Concurrently to us, similar approaches have been proposed by Held et al. [86] and Tao et al. [206]. A brief comparison is provided in Chapter 5, while the review of Pflugfelder [168] presents a more detailed analysis of the differences between these three methods and their successors.

In general, because of the great focus on online learning and the lack of large video datasets, we believe the tracking community has so far overlooked the possibility of tackling the limited data regime by exploiting offline training with similarity learning and meta-learning. In Chapters 5 and 6, we will illustrate our contributions in this rather uncharted territory.

2.2 FEW-SHOT CLASSIFICATION

Arguably, Miller et al. [144] have been among the first to recognise the importance of learning a classifier from few examples and to observe that concepts acquired from previously encountered classes are crucial to perform *fast mapping* [31]. In order to constitute a body of *prior knowledge*, in their work they first learn a probability density function on a space of deformations, which is then used to improve the performance of several optical character classifiers. Fei-Fei et al. [59] adopted a similar approach, but instead of experimenting with the 10 handwritten digits of MNIST, they considered realistic images from 101 different classes. In their case, prior knowledge takes the form of a density function on the parameters of probabilistic models representing object categories. Recognition is then obtained from the posterior density produced by combining this prior knowledge with newly acquired observations.

Other popular approaches to few-shot learning, such as the ones of Salakhutdinov et al. [188] and Lake et al. [122], use a hierarchical Bayesian model to transfer knowledge from the previously learnt concepts to novel categories. Instead, Tommasi and Caputo [211] resorted to a discriminative approach and adopted a modified least-squares SVM model [1] that can learn a set of weights expressing the importance of the already-learnt categories when performing knowledge transfer. Differently, Fink [62] utilised kernelised nearest-neighbours classifiers, while Bart and Ullman [14] proposed a feature replacement method to generalise from previously seen classes, starting from the intuition that a certain feature is likely to be important to classify a novel category if a similar feature has demonstrated to be important for a similar class.

Currently, both machine learning and computer vision communities are assisting to a significant increase of interest in the problem of few-shot learning. The work of Lake et al. [123] has arguably played an essential role in this phenomenon by compiling the Omniglot dataset, which contains 1623 different classes of handwritten digits grouped in 50 alphabets. Together with the dataset, they also introduced a probabilistic programming approach able to surpass human performance. However, this feat was achieved by making use of strokes data as input together with the characters' images. Vinyals et al. [222] proposed a more complex dataset, *mini*ImageNet, containing 60,000 images of 100 classes randomly sampled from ImageNet (and downsampled to 84×84).

In recent literature, the simplest general approach to few-shot classification is to train a similarity function by exposing it to many "matching" tasks [28, 38, 107]. Despite its simplicity, this general strategy is particularly effective and it is at the core of the current state of the art [72, 199, 202, 222]. Interest-

1 The technique has been originally proposed by Orabona et al. [161] as a way to incrementally learn new poses for prosthetic hands.

ingly, Garcia and Bruna [71] interpreted learning as information propagation from labelled to query images and proposed a graph neural network that can generalise matching-based approaches and can be used in both supervised and semi-supervised scenarios.

Many other recent methods tackling the problem of few-shot classification adopt the *meta-learning* paradigm [220] as a way to share common knowledge across episodes and avoid overfitting. For this reason, in Section 2.3 we give a short overview of the most common meta-learning approaches in the literature.

2.3 META-LEARNING: PAST AND PRESENT

Early days. The topic of meta-learning (or learning to learn) became of great importance in the machine learning community several decades ago, with first examples already appearing in the eighties and early nineties [15, 155, 191, 209, 215]. Utgoff [215] developed what arguably is the first method able to dynamically adjust the inductive bias [146] of a learning algorithm, thus implicitly "imposing an ordering" on the elements of its hypothesis space [220]. Later, Bengio et al. [15] interpreted the rule to update the weights of a neural network as a function that is learnable via gradient descent, genetic algorithms or simulated annealing. Another seminal work is the one of Thrun [208, 209], which presents the so-called *lifelong learning* scenario, in which a learning algorithm gradually encounters an ordered sequence of learning problems. Throughout this course, the learner can benefit from re-using the knowledge accumulated during previous tasks. In later work, Thrun and Pratt [209] stated that an algorithm is *learning to learn* if "[...] *its performance at each task improves with experience and with the number of tasks*". This

characterisation has been inspired by Mitchell's definition of a learning algorithm as a computer program whose performance on a task improves with experience [147]. Similarly, in [220], Vilalta & Drissi explained meta-learning as organised in two nested "learning levels". At the base level, an algorithm is confined within a limited hypothesis space while solving a single task. Contrarily, the meta-level can "accrue knowledge" by spanning multiple tasks, so that the hypothesis space at the base level can be adapted effectively.

Meta-networks. Despite its general efficacy, one shortcoming of metric learning for few-shot classification (Section 2.2) is that learning only occurs within the long-time horizon of the external meta-learning loop. While this can clearly spare costly computations, it also prevents these methods from performing adaptation at test-time. A possible way to overcome the lack of adaptability is to train a neural network capable of predicting its own parameters. This technique has been first introduced by Schmidhuber [192, 193] and recently rediscovered by Bertinetto et al. [17] and Munkhdalai and Yu [152], with application to object tracking and few-shot classification. Rebuffi et al. [177] showed that a similar approach can be used to adapt a neural network to different visual domains with respect to the ones it has been trained on.

Gradient descent as a learnable function. Another popular approach to meta-learning is to consider the gradient update rule of SGD as a parametric and learnable function [15] rather than a fixed ad-hoc routine. Younger et al. [240] and Hochreiter et al. [93] observed that, because of the sequential nature of a learning algorithm, a recurrent neural network can be considered as a meta-learning system. They identify LSTMs as particularly apt for the task because of their ability to span long-term dependencies. Andrychowicz et al. [5] and Ravi and Larochelle [174] presented a modern take on this idea, showing benefits on classification and style transfer.

Backpropagating through gradient descent. A recent and promising research direction is the one set by Maclaurin et al. [139] and by the MAML algorithm [63, 64] of Finn et al. Instead of explicitly designing a meta-learner module for learning the update rule, they backpropagate through the very operation of gradient descent to optimise for the hyperparameters or the initial parameters of the learner. However, backpropagation through gradient descent steps is costly, and thus the total number of steps must be kept small.

Memory and attention. To alleviate the drawback of catastrophic forgetting typical of deep neural networks [142], several recent methods [101, 152, 190, 201] make use of memory-augmented models, which can first retain and then access important and previously unseen information associated with newly encountered tasks. While such memory modules store and retrieve information in the long-time range, approaches based on attention like the one of Vinyals et al. [222] are useful to specify the most relevant pieces of knowledge within a task. Mishra et al. [145] complemented soft attention with temporal convolutions, thus allowing the attention mechanism to access information related to past episodes.

3

META-LEARNING WITH DIFFERENTIABLE CLOSED-FORM SOLVERS

In Chapters 1 and 2 we explained why being able to design learning algorithms that can acquire new concepts with few training examples is important. We also introduced the general concept of meta-learning and provided the high-level intuition behind it. In this chapter, we propose a more formal definition and introduce a simple but effective meta-learning method for the problem of few-shot classification.

3.1 A GENERAL META-LEARNING FRAMEWORK

According to widely accepted definitions of learning [146] and meta-learning [220, 222], an algorithm is "learning to learn" if it can improve its average performance with the number of experienced episodes (by progressively modifying its inductive bias). There are two main components in a meta-learning algorithm: a base learner and a meta-learner [220]. The *base learner* works at the level of individual *episodes* (or *tasks*), which in the few-shot scenario correspond to learning problems characterised by having only a small set of labelled training images available. The *meta-learner*, on the other hand, learns from several such episodes in sequence with the goal of improving the performance of the base learner across episodes.

In other words, the goal of meta-learning is to enable a base learning algorithm to adapt to new episodes efficiently by generalising from a set of training episodes $\mathcal{E} \in \mathbb{E}$. $\mathcal{E}$ can be modelled as a probability distribution of example inputs $x \in \mathbb{R}^m$ and outputs $y \in \mathbb{R}^o$, such that we can write $(x, y) \sim \mathcal{E}$.

In the case of few-shot classification, the inputs are represented by few images belonging to different unseen classes, while the outputs are the (episode-specific) class labels. It is important not to confuse the small sets that are used in an episode $\mathcal{E}$ with the super-set $\mathbb{E}$ (such as Omniglot or *mini*ImageNet, Section 3.3.1) from which they are drawn.

Consider a generic feature extractor, such as commonly used pre-trained networks [1] $\phi(x) : \mathbb{R}^m \to \mathbb{R}^e$. Then, a much simpler episode-specific predictor $f(\phi(x) \,|\, w_{\mathcal{E}}) : \mathbb{R}^e \times \mathbb{R}^p \to \mathbb{R}^o$ can be trained to map input embeddings to outputs. The predictor is parameterised by a set of parameters $w_{\mathcal{E}} \in \mathbb{R}^p$, which are specific to the episode $\mathcal{E}$. To train and assess the predictor on one episode, we use training samples $Z_{\mathcal{E}} = \{(x_i, y_i)\} \sim \mathcal{E}$ and test samples $Z'_{\mathcal{E}} = \{(x'_i, y'_i)\} \sim \mathcal{E}$, sampled independently from the distribution $\mathcal{E}$. Then, it is possible to use a learning algorithm Λ to obtain the parameters $w_{\mathcal{E}} = \Lambda(\phi(Z_{\mathcal{E}}))$, where $\phi(Z_{\mathcal{E}}) \triangleq \{(\phi(x_i), y_i)\}$. The quality of the trained predictor is then computed by a loss function $L : \mathbb{R}^o \times \mathbb{R}^o \to \mathbb{R}$, which is evaluated on the test samples $Z'_{\mathcal{E}}$:

$$q(\mathcal{E}) = \frac{1}{|Z'_{\mathcal{E}}|} \sum_{(x,y) \in Z'_{\mathcal{E}}} L\left(f\left(\phi\left(x\right) ; w_{\mathcal{E}}\right), y\right), \quad \text{with } w_{\mathcal{E}} = \Lambda(\phi(Z_{\mathcal{E}})). \tag{3.1}$$

Other than abstracting away the complexities of the learning algorithm as Λ, eq. 3.1 corresponds to the standard train-test protocol commonly employed in machine learning, here applied to a single episode $\mathcal{E}$. However, simply re-

1 Note that in practice we do not use pre-trained networks, but we are able to train them from scratch.

training a predictor for each episode ignores potentially useful knowledge that can be transferred between them.

For this reason, we now take the step of parameterising ϕ and Λ with two sets of *meta-parameters*, ω and ρ, which can aid the training procedure. In particular, ω affects the representation of the input of the base learner algorithm Λ, while ρ corresponds to the hyperparameters of the base learner, which here can be learnt by the meta-learner loop instead of being manually set as it usually happens in a standard training scenario. These meta-parameters will affect the generalisation properties of the learned predictors. This motivates evaluating the result of training on a base test set $Z'_{\mathcal{E}}$ (eq. 3.1). In order to learn ω and ρ, we minimise the loss on base test sets over all episodes $\mathcal{E} \in \mathbb{E}$:

$$\min_{\omega,\rho} \frac{1}{|\mathbb{E}| \cdot |Z'_{\mathcal{E}}|} \sum_{\mathcal{E}\in\mathbb{E}} \sum_{(x,y)\in Z'_{\mathcal{E}}} L\left(f\left(\phi\left(x\,;\,\omega\right)\,;\,w_{\mathcal{E}}\right),\,y\right), \quad \text{with } w_{\mathcal{E}} = \Lambda(\phi(Z_{\mathcal{E}}\,;\,\omega)\,;\,\rho). \tag{3.2}$$

Since eq. 3.2 consists of a composition of non-linear functions, we can leverage the same tools used successfully in deep learning, namely backpropagation and stochastic gradient descent (SGD), to optimise it. The main obstacle is to choose a learning algorithm Λ that is amenable to optimisation with such tools. This means that, in practice, Λ must be quite simple.

Examples of meta-learning algorithms. Using eq. 3.2, it is possible to describe several of the meta-learning methods in the literature, which mostly differ for the choice of Λ. The feature extractor ϕ is typically a standard CNN, whose intermediate layers are trained jointly as ω (and thus are not episode-specific). The last layer represents the linear predictor f, with episode-specific parameters $w_{\mathcal{E}}$. In Siamese networks [28, 38, 107], f is a nearest neighbour classifier (and $w_{\mathcal{E}}{=}\emptyset$), generalised to soft k-means in Ren et al. [178]. The Learnet [17] uses a factorised CNN or MLP to implement Λ, while MAML [64]

implements it using SGD (and furthermore adapts all parameters of the CNN). In addition to learning ω, some works [5, 174] also learn the hyperparameters of the base learner ρ.

Similarities with the multi-task learning paradigm. Our work – and more generally the few-shot learning literature as a whole – is related to the multi-task learning paradigm [32, 209]. In the past, examples of this paradigm have focused on kernel methods (e.g. [6, 56]) and more recently on deep neural networks [183]. However, several crucial differences exist. In terms of setup, multi-task learning methods are trained to solve a fixed set of T tasks (or domains). At test time, the same T tasks or domains are encountered. For instance, the popular Office-Caltech [75] dataset is constructed by considering all the images from 10 classes present in 4 different datasets (the domains). For multi-task learning, the splits span the domains but contain all the 10 classes. Conversely, few-shot learning datasets have splits with disjoint sets of classes (i.e. one split's classes are not contained in other splits). Moreover, only a few examples (*shots*) can be used as training data within one episode, while in multi-task learning this limitation is not present. For this reason, meta-learning methods applied to few-shot learning (e.g. ours, [64, 145, 174, 222]) crucially take into account adaptation *already during the training process* to mimic the test time setting, de facto learning how to learn from limited data.

3.2 METHOD

Clearly, in any meta-learning algorithm, it is of paramount importance to choose the base learner carefully. On one side of the spectrum, methods related to nearest neighbours, such as learning similarity functions [107, 199,

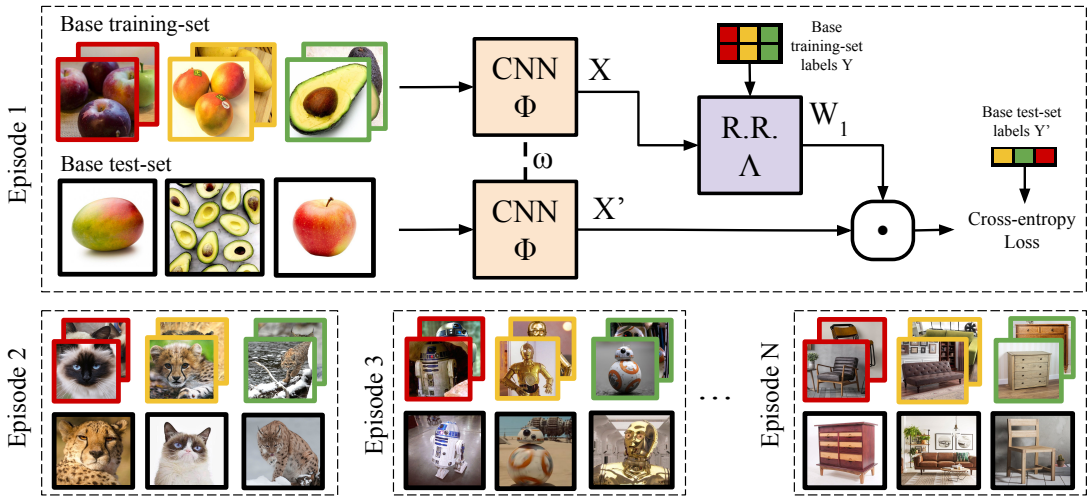


Figure 3.1: Diagram of the proposed method for one episode, of which several are seen during meta-training. The task is to learn *new* classes given just a few sample images per class. In this illustrative example, there are 3 classes and 2 samples per class, making each episode a 3-way, 2-shot classification problem (different labels are represented by different colours). At the base learning level, learning is accomplished by a differentiable ridge regression layer (R.R.), which computes episode-specific weights W. At the meta-training level, by backpropagating errors through many of these small learning problems, we train a network whose weights are shared across episodes, together with the hyperparameters of the R.R. layer. In this way, the R.R. base learner can improve its learning capabilities as the number of experienced episodes increases.

222] and learning how to access a memory module [101, 152, 190], are fast but solely rely on the quality of the similarity metric, with no additional data-dependent adaptation at test-time. On the other side of the spectrum, methods that optimise standard iterative learning algorithms, such as backpropagating through gradient descent [64, 157] or explicitly learning the learner's update rule [5, 93, 174], are slower but allow more adaptability to different problems or domains.

In this paper, we take a different perspective. As base learner Λ, we adopt Ridge Regression: a simple learning algorithm that admits a closed-form solution. Crucially, the simplicity and differentiability of its solution allow us to backpropagate through learning problems, achieving meta-learning. Simple algorithms like Ridge Regression are particularly suitable for use within a meta-learning framework for few-shot classification for two reasons. First,

their closed-form solution allows learning problems to be solved efficiently. Second, in a data regime characterised by few examples of high dimensionality, the Woodbury's identity [167, Chapter 3.2] can be used to obtain a very significant gain in terms of computational speed.

In the outer meta-training loop, we allow SGD to learn both the parameters ω of the feature representation of Λ and its hyperparameters ρ.

We demonstrate the strength of our approach by performing extensive experiments on Omniglot [123], CIFAR-100 [116] (adapted to the few-shot problem) and *mini*ImageNet [222]. Our base learners are fast, simple to implement, and can achieve a performance that is competitive with or superior to the state of the art in terms of accuracy.

3.2.1 *R2-D2: efficient ridge regression base learners*

Similarly to the methods discussed in Section 3.1, over the course of a *single* episode we adapt a linear predictor f, which can be considered as the final layer of a CNN. The remaining layers ϕ are trained from scratch to generalise between episodes by the outer meta-learning loop (eq. 3.2), but for the purposes of one episode they are considered fixed. In this section we assume that the inputs were pre-processed by the CNN ϕ, and that we are dealing only with the final linear predictor $f(\phi(x)) = \phi(x)W \in \mathbb{R}^o$, where the parameters $w_{\mathcal{E}}$ are reorganized into a matrix $W \in \mathbb{R}^{e \times o}$.

The motivation for our work is that, while not quite as simple as nearest neighbours, least-squares regressors admit closed-form solutions. Although simple least squares is prone to overfitting, it is easy to augment it with L^2

regularization (controlled by a positive hyperparameter λ), in what is known as ridge regression:

$$\Lambda(Z) = \arg\min_W \|XW - Y\|^2 + \lambda \|W\|^2 \tag{3.3}$$

$$= (X^T X + \lambda I_{e,e})^{-1} X^T Y, \tag{3.4}$$

where $X \in \mathbb{R}^{n \times e}$ and $Y \in \mathbb{R}^{n \times o}$ contain the n sample pairs of input embeddings and outputs from Z, stacked as rows.

Because ridge regression admits a closed-form solution (eq. 3.4), it is relatively easy to integrate into meta-learning (eq. 3.2) using standard automatic differentiation packages. The only element that may have to be treated more carefully is the matrix inversion. When the matrix to invert is close to singular (which is not the case when $\lambda > 0$), it is possible to achieve more numerically accurate results by replacing the matrix inverse and vector product with a linear system solver [153, Section 7.5.2].

Another concern about eq. 3.4 is that the size of the intermediate matrix $X^T X \in \mathbb{R}^{e \times e}$ grows *quadratically* with the embedding size e. Given the high dimensionality of features typically used in deep networks, the inversion could come at a very expensive cost. To alleviate this, we rely on the Woodbury formula [167, Section 3.2], obtaining:

$$W = \Lambda(Z) = X^T (XX^T + \lambda I_{n,n})^{-1} Y. \tag{3.5}$$

The main difference between eq. 3.4 and eq. 3.5 is that the intermediate matrix $XX^T \in \mathbb{R}^{n \times n}$ grows quadratically with the number of samples in the episode, n. As we are interested in one-shot or few-shot learning, this is typically very small. The overall cost of eq. 3.5 is only linear in the embedding size e.

Although this method was originally designed for regression, we found that it works well also in a (few-shot) classification scenario, where the target outputs are one-hot vectors representing classes. However, since eq. 3.4 does not directly produce classification labels, it is important to calibrate its output for the cross-entropy loss (L in eq. 3.2), which is used to evaluate one episode's test samples. This can be done by simply adjusting the prediction $X'W$ with a scale and a bias $\alpha, \beta \in \mathbb{R}$:

$$\widehat{Y} = \alpha X'W + \beta. \tag{3.6}$$

Note that λ, α and β are hyperparameters of the base learner Λ and can be learnt by the outer learning loop together with the CNN parameters ω.

3.2.2 *LR-D2: iterative base learners and logistic regression*

It is natural to ask whether other learning algorithms can be integrated as efficiently as ridge regression within the meta-learning framework described in Section 3.1. In general, a similar derivation is possible for iterative solvers, as long as the operations are differentiable. For linear models with convex loss functions, a better choice than gradient descent is Newton's method, which uses curvature (second-order) information to obtain the solution in very few steps. One learning objective of particular interest is logistic regression, which unlike ridge regression directly produces classification labels and thus does not require the use of calibration before the (binary) cross-entropy loss.

When one applies Newton's method to logistic regression, the resulting algorithm takes a familiar form – it consists of a series of weighted least squares (or ridge regression) problems, giving it the name Iteratively Reweighted

Least Squares (IRLS) [153, Section 8.3.4]. Given inputs $X \in \mathbb{R}^{n \times e}$ and binary outputs $y \in \{-1, 1\}^n$, the i-th iteration updates the parameters $w_i \in \mathbb{R}^e$ as:

$$w_i = \left(X^T \text{diag}(s_i) X + \lambda I_{e,e}\right)^{-1} X^T \text{diag}(s_i) z_i, \tag{3.7}$$

where I is an identity matrix, $s_i = \mu_i(1 - \mu_i)$, $z_i = w_{i-1}^T X + (y - \mu_i)/s_i$, and $\mu_i = \sigma(w_{i-1}^T X)$ applies a sigmoid function σ to the predictions using the previous iteration's parameters w_{i-1}.

Since eq. 3.7 takes a similar form to ridge regression, we can use it for meta-learning in the same way as we did in Section 3.2.1, with the difference that a small number of steps (eq. 3.7) must be performed in order to obtain the final parameters $w_{\mathcal{E}}$. Similarly, we obtain a solution with a cost which is linear rather than quadratic in the embedding size by employing the Woodbury formula:

$$w_i = X^T \left(XX^T + \lambda \text{diag}(s_i)^{-1}\right)^{-1} z_i, \tag{3.8}$$

where the inner inverse has negligible cost since it is a diagonal matrix. Note that a similar strategy could be followed for other learning algorithms, such as L^1 minimization, LASSO and multinomial logistic regression. We consider logistic regression an illustrative example, of particular interest for binary classification in one/few-shot learning, leaving the exploration of other variants for future work.

3.2.3 *Training policy*

Figure 3.1 illustrates our overall framework. Like most meta-learning techniques, we organize our training procedure into *episodes*, each of which corresponds to a few-shot classification problem. In standard classification, train-

ing requires sampling from a distribution of images and labels. Instead, in our case, we sample from a distribution of episodes, each containing its own training set and test set with just few samples per image. Each episode also contains two sets of labels: Y and Y'. The former is used to train the base learner, while the latter serves to compute the error of the just-trained base learner, enabling the learning of ω, λ, α and β via backpropagation.

In our implementation, one episode corresponds to an SGD mini-batch of size $S = N(K + Q)$, where N is the number of different classes ("ways"), K is the number of samples per class ("shots") and Q is the number of query (or test) images per class.

3.3 EXPERIMENTS

In this section, we provide practical details for the two novel methods introduced in Sections 3.2.1 and 3.2.2, which we dub R2-D2 (*Ridge Regression Differentiable Discriminator*) and LR-D2 (*Logistic Regression Differentiable Discriminator*). We analyze their performance against the recent literature using three few-shot learning benchmarks: Omniglot [123], *mini*ImageNet [222] and CIFAR-FS, which we introduce.

3.3.1 *Few-shot learning benchmarks*

Let $I_\star$ and $C_\star$ be respectively the set of images and the set of classes belonging to a certain data split $\star$. In standard classification datasets, $I_{\text{train}} \cap I_{\text{test}} = \varnothing$ and $C_{\text{train}} = C_{\text{test}}$. Instead, the few-shot setup requires both $I_{\text{meta-train}} \cap I_{\text{meta-test}} = \varnothing$ and $C_{\text{meta-train}} \cap C_{\text{meta-test}} = \varnothing$, while within an episode we have $C_{\text{task-train}} = C_{\text{task-test}}$.

Omniglot [123] is a dataset of handwritten characters that has been referred to as the "MNIST transpose" for the high number of classes and the small number of instances per class. It contains 20 examples of 1623 characters, grouped in 50 different alphabets. In order to be able to compare against the state of the art, we adopt the same setup and data split used in Vinyals et al. [222]. Hence, we resize images to 28×28 and we augment the dataset using four rotated versions of the each instance (0°, 90°, 180°, 270°). Including rotations, we use 4800 classes for meta-training and meta-validation and 1692 for meta-testing.

***mini*ImageNet** [222] aims at representing a challenging dataset without demanding considerable computational resources. It is randomly sampled from ImageNet [185] and it is constituted by a total of 60,000 images from 100 different classes, each with 600 instances. All images are RGB and have been downsampled to 84×84. As all recent work, we adopt the same splits of Ravi and Larochelle [174], who employ 64 classes for meta-training, 16 for meta-validation and 20 for meta-testing.

CIFAR-fs. On the one hand, despite being lightweight, Omniglot is becoming too simple for modern few-shot learning methods, especially with the splits of Vinyals et al. [222]. On the other, *mini*ImageNet is more challenging, but it might still require a model to train for several hours before convergence. Thus, we propose CIFAR-FS (CIFAR100 few-shots), which is randomly sampled from CIFAR-100 [116] by using the same criteria with which *mini*ImageNet has been generated. We observed that the average inter-class similarity is sufficiently high to represent a challenge for the current state of the art. Moreover, the limited original resolution of 32×32 makes the task harder and at the same time allows fast prototyping. The classes contained in each split are detailed in Appendix A.3.

3.3.2 *Experimental results*

In order to produce the features X for the base learners (eq. 3.4 and 3.7), as many recent methods we use a shallow network of four convolutional "blocks", each consisting of the following sequence: a 3×3 convolution (padding=1, stride=1), batch-normalization, 2×2 max-pooling, and a leaky-ReLU with a factor of 0.1. Max pooling's stride is 2 for the first three layers and 1 for the last one. The four convolutional layers have $[96, 192, 384, 512]$ filters. Dropout is applied to the last two blocks for the experiments on *mini*ImageNet and CIFAR-FS, respectively with probabilities 0.1 and 0.4. We do not use any fully-connected layer. Instead, we flatten and concatenate the output of the third *and* fourth convolutional blocks and feed it to the base learner. Doing so, we obtain high-dimensional features of size 3584, 72576 and 8064 for Omniglot, *mini*ImageNet and CIFAR-FS respectively. It is important to mention that the use of the Woodbury formula (section 3.2.1) allows us to make use of high-dimensional features without incurring burdensome computations. In fact, in few-shot problems the data matrix X is particularly "large and short". As an example, with a 5-way/1-shot problem from *mini*ImageNet we have $X \in \mathbb{R}^{5\times72576}$. Applying the Woodbury identity, we obtain significant gains in computation, as in eq. 3.5 we invert a matrix that is only 5×5 instead of 72576×72576.

As Snell et al. [199], we observe that using a higher number of classes during training is important. Hence, despite the few-shot problem at test time being 5 or 20-way, in our multi-class classification experiments we train using 60 classes for Omniglot, 16 for *mini*ImageNet and 20 for CIFAR-FS. Moreover, in order not to train a different model for every single configuration (two for *mini*ImageNet and CIFAR-FS, four for Omniglot), similarly to [145] and differently from previous work, we train our models with a random number

of shots, which does not deteriorate the performance and allow us to simply train one model per dataset. We then choose Q (the size of the query or test set) accordingly, so that the batch size S remains constant throughout the episodes. We set S to 600 for Omniglot and 240 for both *mini*ImageNet and CIFAR-FS.

At the meta-learning level, we train our methods with Adam [106] with an initial learning rate of 0.005, dampened by 0.5 every 2,000 episodes. Training is stopped when the error on the meta-validation set does not decrease meaningfully for 20,000 episodes.

As for the base learner, we let the optimiser learn the parameters ω of the CNN, as well as the regularisation factor λ and the scale α and bias β of the calibration layer of R2-D2 (end of Section 3.2.1). In practice, we observed that it is important to use SGD to adapt α and β, while it is indifferent whether λ is learnt or not. A more detailed analysis can be found in Appendix A.2.

Multi-class classification. Tables 3.1 and 3.2 show the performance of our closed-form base learner R2-D2 against the current state of the art for shallow architectures of four convolutional layers. Values represent average classification accuracies obtained by sampling 10,000 episodes from the meta test-set and are presented with 95% confidence intervals. For each column, the best performance is in bold. If more than one value is outlined, it means their intervals overlap. For prototypical networks, we report the results reproduced by the code provided by the authors. For our comparison, we report the results of methods which train their models from scratch for few-shot classification, omitting very recent work of Qiao et al. [173] and Gidaris and Komodakis [72], which instead make use of pre-trained embeddings.

In terms of feature embeddings, Finn et al. [64], Ravi and Larochelle [174], Snell et al. [199], Vinyals et al. [222] use 64 filters per layer (which become 32 for *mini*ImageNet in [174] and [64] to limit overfitting). On top of this,

Sung et al. [202] also uses a *relation module* of two convolutional and two fully-connected layers. GNN [71] employs an embedding with $[64, 96, 128, 256]$ filters, a fully-connected layer and a graph neural network (with its own extra parameters). To ensure a fair comparison, we increased the capacity of the architectures of three representative methods (MAML, prototypical networks and GNN) to match ours. The results of these experiments are reported with a $*$ on Table 3.1. We make use of dropout on the last two layers for all the experiments on baselines with $*$, as we verified it is helpful to reduce overfitting. Moreover, we report results for experiments on our R2-D2 in which we use a 64-channels embedding.

Despite its simplicity, our proposed method achieves an average accuracy that, on *mini*ImageNet and CIFAR-FS, is superior to the state of the art with shallow architectures. For example, on the four problems of Table 3.1, R2-D2 improves on average of a relative 4.3% w.r.t. GNN (the second-best method). R2-D2 shows competitive results also on Omniglot (Table 3.2), achieving among the best performance for all problems. Furthermore, when we use the "lighter" embedding, we can still observe a performance which is in line with the state of the art. Interestingly, increasing the capacity of the other methods is not particularly helpful. It is beneficial only for GNN on *mini*ImageNet and prototypical networks on CIFAR-FS, while being detrimental in all the other cases.

Our R2-D2 is also competitive against SNAIL, which uses a much deeper architecture (a ResNet with a total of 14 convolutional layers). Despite being outperformed for the 1-shot case, we can match its results on the 5-shot one. Moreover, it is paramount for SNAIL to make use of such deep embedding, as its performance drops significantly with a shallow one.

LR-D2 performance on multi-class classification. In order to be able to compare our binary classifier LR-D2 with the state-of-the-art in few-shot N-class classification, it is possible to jointly consider N binary classifiers, each of

Table 3.1: Few-shot multi-class classification accuracies on *mini*ImageNet and CIFAR-FS.

Method	***mini*ImageNet, 5-way** 1-shot	5-shot	**cifar-fs, 5-way** 1-shot	5-shot
Matching net [222]	44.2%	57%	—	—
MAML [64]	48.7±1.8%	63.1±0.9%	58.9±1.9%	71.5±1.0%
MAML ∗	40.9±1.5%	58.9±0.9%	53.8±1.8%	67.6±1.0%
Meta-LSTM [174]	43.4±0.8%	60.6±0.7%	—	—
Proto net [199]	47.4±0.6%	65.4±0.5%	55.5±0.7%	72.0±0.6%
Proto net ∗	42.9±0.6%	65.9±0.6%	57.9±0.8%	76.7±0.6%
Relation net [202]	50.4±0.8%	65.3±0.7%	55.0±1.0%	69.3±0.8%
SNAIL (with *ResNet*) [145]	*55.7±1.0%*	*68.9±0.9%*	—	—
SNAIL (with 32C) [145]	45.1%	55.2%	—	—
GNN [71]	50.3%	66.4%	61.9%	75.3%
GNN∗	50.3%	**68.2%**	56.0%	72.5%
Ours/R2-D2 (with 64C)	49.5±0.2%	65.4±0.2%	**62.3**±0.2%	**77.4**±0.2%
Ours/R2-D2	**51.8**±0.2%	**68.4**±0.2%	**65.4**±0.2%	**79.4**±0.2%
Ours/LR-D2 (1 iter.)	51.0±0.2%	65.6±0.2%	**64.5**±0.2%	75.8±0.2%
Ours/LR-D2 (5 iter.)	**51.9**±0.2%	**68.7**±0.2%	**65.3**±0.2%	**78.3**±0.2%

Table 3.2: Few-shot multi-class classification accuracies on Omniglot.

Method	**Omniglot, 5-way** 1-shot	5-shot	**Omniglot, 20-way** 1-shot	5-shot
Siamese net [107]	96.7%	98.4%	88%	96.5%
Matching net [222]	98.1%	98.9%	93.8%	98.5%
MAML [64]	**98.7**±0.4%	**99.9**±0.1%	95.8±0.3%	98.9±0.2%
Proto net [199]	98.5±0.2%	99.5±0.1%	95.3±0.2%	98.7±0.1%
SNAIL [145]	**99.07**±0.16%	**99.77**±0.09%	**97.64**±0.30%	**99.36**±0.18%
GNN [71]	**99.2%**	**99.7%**	**97.4%**	99.0%
Ours/R2-D2 (with 64C)	98.55±0.05%	99.66±0.02%	94.70±0.05%	98.91±0.02%
Ours/R2-D2	**98.91**±0.05%	**99.74**±0.02%	96.24±0.05%	**99.20**±0.02%

which discriminates between a specific class and all the remaining ones [22, Chapter 4.1]. In our framework, this can be easily implemented by concatenating together the outputs of N instances of LR-D2, resulting in a single multi-class prediction.

We use the same setup and hyper-parameters of R2-D2, except for the number of classes/ways used at training, which we limit to 10. Interestingly, with five IRLS iterations the accuracy of the 1-vs-all variant of LR-D2 is similar to the one of R2-D2 (Table 3.1): 51.9% and 68.7% for *mini*ImageNet (1-shot and

Table 3.3: Few-shot binary classification accuracies on *mini*ImageNet and CIFAR-FS.

Method	***mini*ImageNet, 2-way** 1-shot	5-shot	**cifar-fs, 2-way** 1-shot	5-shot
MAML [64]	74.9±3.0%	84.4±1.2%	82.8±2.7%	88.3±1.1%
Proto nets [199]	71.7±1.0%	84.8±0.7%	76.4±0.9%	88.5±0.6%
Relation net [202]	76.2±1.2%	**86.8**±1.0%	75.0±1.5%	86.7±0.9%
GNN [71]	**78.4%**	**87.1%**	79.3%	89.1%
Ours/R2-D2	77.4±0.3%	**86.8**±0.2%	**84.1**±0.3%	**91.7**±0.2%
Ours/LR-D2 (10 iter.)	**78.1**±0.3%	**86.5**±0.2%	**84.7**±0.3%	**91.5**±0.2%

5-shot); 65.3% and 78.3% for CIFAR-FS. With a single iteration, performance is still very competitive: 51.0% and 65.6% for *mini*ImageNet; 64.5% and 75.8% for CIFAR-FS. However, the requirement of solving N binary problems per iteration makes it much less efficient than R2-D2, as evident in Table 3.4.

Binary classification. Finally, in Table 3.3 we report the performance of both our ridge regression and logistic regression base learners, together with four representative methods. Since LR-D2 is limited to operate in a binary classification setup, we run our R2-D2 and prototypical network without oversampling the number of ways. For both methods and prototypical networks, we report the performance obtained annealing the learning rate by a factor of 0.99, which works better than the schedule used for multi-class classification. Moreover, motivated by the small size of the mini-batches, we replace Batch Normalization with Group Normalization [234]. For this table, we use the default setup found in the code of MAML, which uses 5 SGD iterations during training and 10 during testing. Table 3.3 confirms the validity of both our approaches on the binary classification problem.

Although different in nature, both MAML and our LR-D2 make use of iterative base learners: the former is based on SGD, while the latter on Newton's method (under the form of Iteratively Reweighted Least Squares). The use of second-order optimisation might suggest that LR-D2 is characterized by computationally demanding steps. However, we can apply the Woodbury identity

at every iteration and obtain a significant speedup. In Figure 3.2 we compare the performance of LR-D2 vs the one of MAML for a different number of steps of the base learner (kept constant between training and testing). LR-D2 is superior to MAML, especially for a higher number of steps.

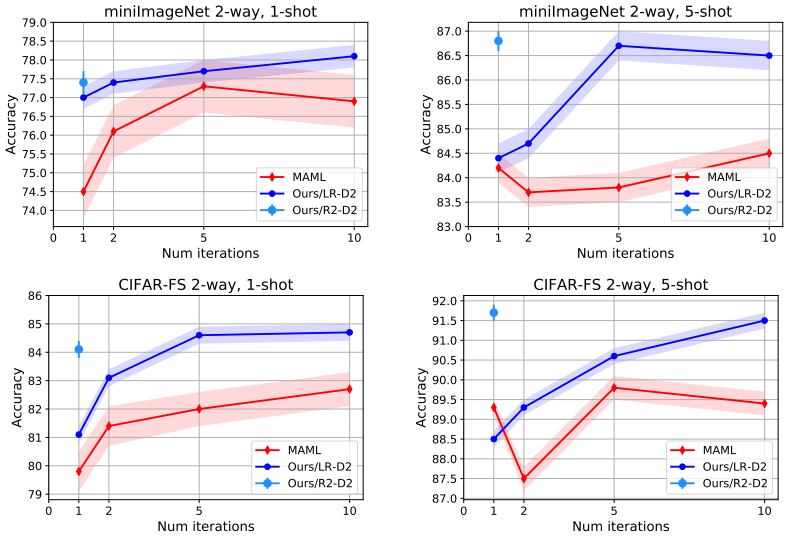


Figure 3.2: Binary classification accuracy on two datasets and two setups at different number of steps of the base learner for MAML, R2-D2 and LR-D2. Shaded areas represent 95% confidence intervals.

Efficiency. In Table 3.4 we compare the amount of time required by two representative methods and ours to solve 10,000 episodes (each with 10 images) on a single NVIDIA GTX 1080 GPU. We use *mini*ImageNet (5-way, 1-shot) and adopt, for the lower part of the table, a lightweight embedding network of 4 layers and 32 channels per layer. For reference, in the upper part of the table we also report the timings for R2-D2 with $[64, 64, 64, 64]$ and $[96, 192, 384, 512]$ embeddings.

Interestingly, we can observe how R2-D2 allows us to achieve an efficiency that is comparable to the one of prototypical networks and significantly higher than MAML. Notably, unlike prototypical networks, our methods do allow per-episode adaptation through the weights W of the solver.

Table 3.4: Time required to solve 10,000 *mini*ImageNet episodes of 10 samples each.

	***mini*ImageNet, 5-way, 1-shot**
Ours/R2-D2	1 min 23 sec
Ours/R2-D2 (with 64C)	1 min 4 sec
MAML [64] (with 32C)	6 min 35 sec
Ours/LR-D2 (1-vs-all) (1 iter.) (with 32C)	5 min 48 sec
Ours/R2-D2 (with 32C)	57 sec
Proto nets [199] (with 32C)	24 sec

3.4 DISCUSSION

In this work, we proposed two novel methods to tackle the few-shot learning scenario, which consists in learning how to discriminate between images given one or very few examples. For methods tackling this problem, it is common practice to organise the training procedure in two *nested loops*. Methods in the recent literature mostly differ for the type of learner they use in the inner loop and the amount of per-episode adaptability they allow (Section 3.1). For example, at the one end of the spectrum in terms of "amount of adaptability", we can find methods such as MAML [64], which learns how to efficiently fine-tune the parameters of a neural-network with few iterations of SGD. On the other end, we have methods based on metric learning such as prototypical networks [199] and relation network [202], which are fast but do not perform adaptation. Note that the amount of adaptation to a new episode (i.e. a new classification problem with unseen classes) is not at all indicative of the performance in few-shot learning benchmarks. As a matter of fact, both Snell et al. [199] and Sung et al. [202] achieve higher accuracy than MAML. Nonetheless, adaptability is a desirable property, as it allows more design flexibility.

Within this landscape, our work proposes a novel technique (R2-D2) that does allow per-episode adaptation while at the same time being fast (Table 3.4) and achieving strong performance (Table 3.1). The key innovation is to use a

simple (and differentiable) solver such as ridge regression within the inner loop, which requires *backpropagating through the solution of a learning problem*. Crucially, its closed-form solution and the use of the Woodbury identity (particularly advantageous in the low-data regime) allow this non-trivial endeavour to be efficient. Furthermore, we demonstrated that this strategy is not limited to the ridge regression case, but it can also be extended to other solvers (LR-D2) by dividing the problem into a short series of weighted least squares problems. We showed that these differentiable learning blocks work remarkably well, with excellent results on three few-shot learning benchmarks.

We believe that our findings point in an exciting direction of more sophisticated online adaptation methods, able to leverage the potential of prior knowledge distilled in an offline training phase.

4

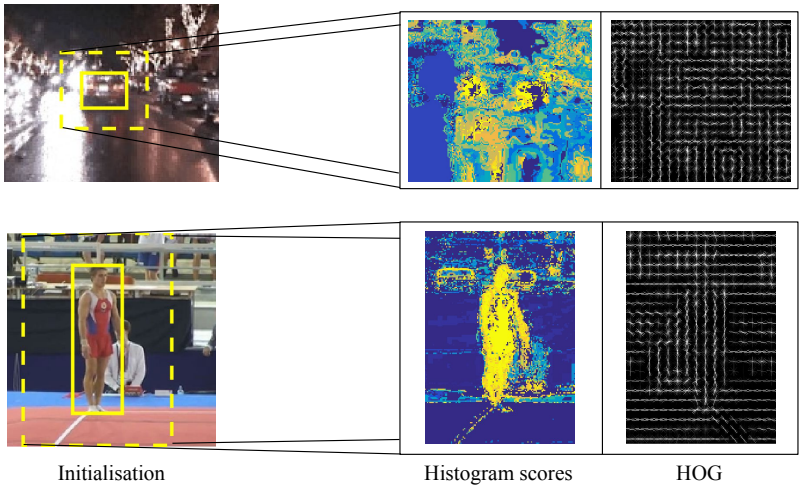
STAPLE: COMPLEMENTARY LEARNERS FOR REAL-TIME TRACKING

4.1 INTRODUCTION

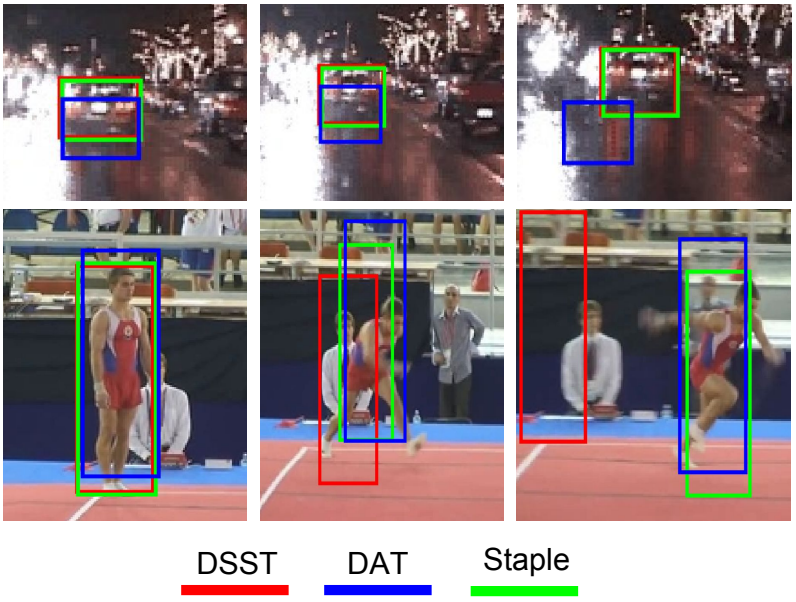
In this chapter and the following two, we consider the scenario (described in Section 2.1) of short-term, causal, single-object tracking, in which the target is only specified in the first frame using a rectangle. The key challenge is to be robust to changes in its appearance, while only having access to one labelled example. Moreover, it is desirable for a tracking algorithm to operate above real-time speed to satisfy the needs of computationally intensive applications such as robotics, surveillance, video processing and augmented reality.

Since object appearance can vary significantly during a video, it is not generally sufficient to estimate its model from the first frame alone and use this single, fixed model to locate the object in all subsequent frames. Most trackers (e.g. [83, 88, 102]), therefore, employ model adaptation to take advantage of the information present in later frames. The most common approach is to treat the tracker's predictions in new frames as training data with which to update the model. However, the danger of learning from predictions is that small errors can accumulate and cause model drift.

In this chapter, we propose *Staple* (**S**um of **t**emplate **a**nd **P**ixel-wise **le**arners), a tracker that combines two object representations that are sensitive to complementary factors to learn a model that is inherently robust to both colour



(a)



(b)

Figure 4.1: Color distributions can be inadequate to discriminate the target from the background (top row of (a) and (b)). In contrast, template models depend on the spatial configuration of the object (bottom row of (a) and (b)) and thus perform poorly when this changes rapidly. *Staple* can rely on the strengths of both template and colour-based models. Its performance is not affected by non-distinctive colours (unlike DAT [171]) and, at the same time, it is robust to deformations (unlike DSST [42]).

changes and object deformations. To achieve a high speed, we solve two independent ridge-regression problems, exploiting the inherent structure of each representation. Compared to other algorithms that fuse the *predictions* of mul-

tiple models, our tracker combines the *scores* of two models in a dense translation search, enabling greater accuracy. A critical property of the two models is that their scores are similar in magnitude and indicative of their reliability, so that the prediction is dominated by the more confident.

We establish the surprising result that a basic tracker that simply combines a Correlation Filter (using HOG features) and a global colour histogram outperforms several more complex methods in multiple benchmarks, while running at 80 frames per second.

For a scheme illustrating the intuition behind the proposed method, see Figure 4.1.

4.2 RELATED WORK

Online learning and the Correlation Filter. As discussed in Section 2.1, many approaches tackling arbitrary object tracking use an online version of an object detection algorithm which is initialised in the first frame and updated in the following ones. When collecting (online) positive and negative samples from the video frames to train the object detector, most trackers are constrained by design to use random crops, a strategy which is inefficient because it does not exploit the redundancies between overlapping windows.

Conversely, the Correlation Filter [118] is an algorithm that minimises a (least-squares) loss for *all* circular shifts of the positive example. Although this might seem a weaker approximation of the true problem, it enables the use of densely-sampled examples and high-dimensional feature images in real-time using the Fourier domain.

Initially applied to arbitrary object tracking on grayscale images by Bolme et al. [25], its extension to multiple feature channels [24, 88, 90, 104] and there-

fore HOG features [41] enabled the technique to achieve state-of-the-art performance in VOT-2014 [109]. The winner of the challenge, DSST [42], incorporated a multi-scale template for Discriminative Scale-Space Tracking using a 1D Correlation Filter.

One shortcoming of Correlation Filter-based trackers is that they are constrained to learn from *all* circular shifts. Several recent works [45, 61, 105] have sought to resolve this issue, and the Spatially Regularised (SRDCF) [45] formulation, in particular, has demonstrated excellent tracking results. However, this was achieved at the cost of real-time operation.

Robustness to deformation. Correlation Filters are inherently confined to the problem of learning a rigid template. This limitation is a concern, as it is common for arbitrary targets to undergo shape deformations in the course of a sequence. Perhaps the simplest method to achieve robustness to deformation is to adopt a representation that is insensitive to shape variation. Image histograms have this property because they discard the position of every pixel. In fact, histograms can be considered "orthogonal" to Correlation Filters, since a Correlation Filter is *learnt from* circular shifts, whereas a histogram is *invariant to* circular shifts.

However, histograms alone are often insufficient to discriminate the object from the background. While colour histograms were used in many early approaches to object tracking [159, 166], they have only recently demonstrated good performance in modern benchmarks with the Distractor-Aware Tracker (DAT) [171], which uses adaptive thresholding and explicit suppression of regions with similar colours. In general, histograms may be constructed from any discrete-valued feature, including local binary patterns and quantised colours. For a histogram to provide robustness to deformation, the feature must be insensitive to the local changes that arise.

The chief alternative way to achieve robustness to deformation is to learn a deformable model. Nonetheless, we believe it is ambitious to learn a deformable model from a single video in which the only supervision is the location in the first frame, besides expressed as a simple rectangular bounding box. While our method outperforms recent sophisticated parts-based models [30, 235] in benchmarks, deformable models have a richer representation that these evaluations do not necessarily reward.

Rather than use a deformable model, HoughTrack [73] and PixelTrack [55] accumulate votes from each pixel and then use the pixels that voted for the winning location to estimate the object's extent. However, these methods are yet to demonstrate competitive benchmark performances.

Strategies to reduce model drift. Model drift is a result of learning from inaccurate predictions. Several works have aimed to prevent drift by modifying the training strategy rather than improving the predictions. TLD [102] and PROST [189] encode rules for additional supervision based on optical flow and a conservative appearance model. Other approaches avoid or delay making hard decisions. MILTrack [10] uses Multiple-Instance Learning to train with bags of positive examples. Supančič and Ramanan [203] introduced self-paced learning for tracking. Their method solves for the optimal trajectory keeping the appearance model, then updates the model using the most confident frames, and repeat. In [79], Grabner et al. treated tracking as online semi-supervised boosting, in which a classifier learnt in the first frame provides an anchor for the labels assigned to examples in later frames. Tang et al. [205] applied co-training to tracking, learning two independent SVMs that use different features and then obtaining hard negatives from the combined scores.

Of these methods, only MILTrack and TLD are found in current benchmarks, and neither has strong results.

Combining multiple estimates. Another strategy widely adopted to mitigate inaccurate predictions is to combine the estimates of an *ensemble* of methods so that the weaknesses of individual trackers are reciprocally compensated. In [120, 121], Kwon et al. suggested to use complementary basic trackers, built by combining different observation models and motion models, and then integrate their estimates in a sampling framework. Similarly, Wang and Yeung [226] combined five independent trackers using a factorial HMM, modelling both the object trajectory and the reliability of each tracker across time. Rather than using trackers of different types, the Multi-Expert Entropy Minimisation (MEEM) tracker [244] maintains a collection of past models and chooses the prediction of one of them according to an entropy criterion.

We differ from these approaches in that *a)* both of our models are learnt in a common framework (specifically, ridge regression), and *b)* this enables us to directly combine the scores of the two models in a dense search.

Long-term tracking with re-detection. Several works have adopted the Correlation Filter for the problem of long-term tracking [134, 148, 219], where it is essential to have the capability of re-detecting the target object. The Long-term Correlation Tracker (LCT) [137] augments a standard Correlation Filter tracker with an additional Correlation Filter for confidence estimation and a random forest for re-detection. The Multi-Store Tracker (MUSTer) [95] maintains a long-term memory of SIFT keypoints for the object and background, using keypoint matching and MLESAC [212] to locate the object. The confidence of the long-term memory is estimated using the number of inliers, and occlusions can be determined by considering the number of background keypoints that are located inside the rectangle. Note that the TLD [102] and self-paced learning [203] algorithms also incorporate some aspects that are well-suited to the long-term tracking problem. A comparison between TLD, LCT and *Staple* in a long-term tracking scenario is presented in [219].

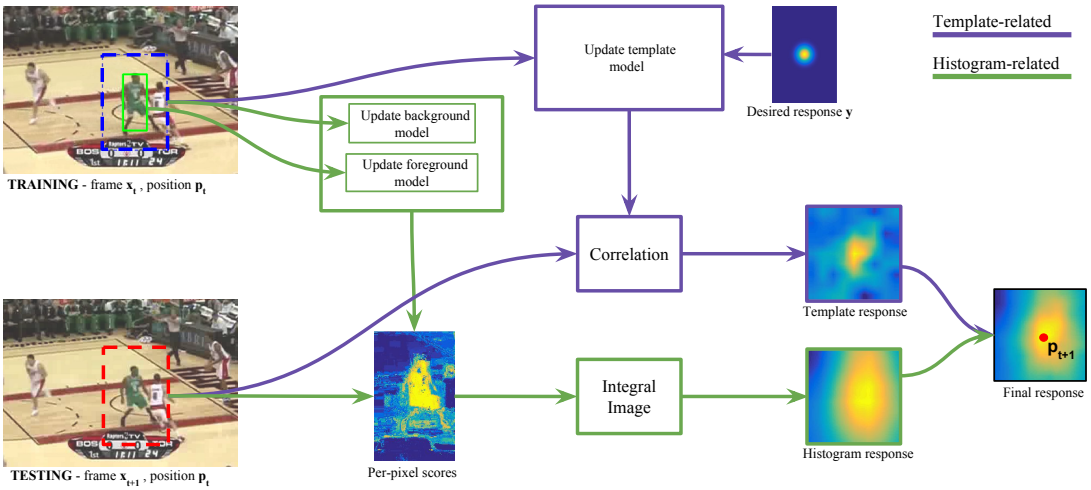


Figure 4.2: Template. For each frame x_t, a training patch is extracted at the (previously estimated) location p_t and used to update the template model in eq. 4.16. At $t+1$, a *test* patch is extracted around the latest estimated location (p_t) and combined with the template model to obtain a dense template response. **Histogram.** At each frame x_t, foreground and background regions are used to update the colour bins in eq. 4.21. These statistics are used to update the weights β_t in eq 4.20. At x_{t+1}, a per-pixel score is computed in a search area centred at p_t, which is then used to obtain the dense histogram response efficiently using an integral image in eq. 4.8. The final response is obtained by linearly combining the two dense responses in eq. 4.4. The new location p_{t+1} of the target is estimated at its peak.

4.3 METHOD

4.3.1 *General learning framework*

During tracking, at frame t, the rectangle p_t that gives the target location in image x_t is chosen from a set $\mathcal{S}_t$ to maximise a score:

$$p_t = \arg\max_{p \in \mathcal{S}_t} f\big(T(x_t, p); \theta_{t-1}\big) \ . \tag{4.1}$$

T selects the rectangle p from x such that $f(T(x,p);\theta)$ assigns a score to the rectangular window p in image x according to the model parameters θ. Ideally, the model parameters should be chosen to minimise a loss function $L(\theta; \mathcal{X}_t)$

that directly depends on all the previous images and the location[1] of the object in those images $\mathcal{X}_t = \{(x_i, p_i)\}_{i=1}^{t}$

$$\theta_t = \arg\min_\theta L(\theta; \mathcal{X}_t). \tag{4.2}$$

In practice, to achieve real-time performance, the functions f and L are chosen not only to locate the object reliably and accurately, but also taking into account that the problems in eq. 4.1 and eq. 4.2 should be solved efficiently. For this reason, the training loss that we optimise to choose the parameters is a weighted linear combination of per-image losses. More specifically, similarly to popular Correlation Filter-based trackers [25, 42, 88], we adopt a recursive definition of the loss function with adaptivity rate η:

$$\widetilde{L}(\theta; \mathcal{X}_t) = (1 - \eta)\widetilde{L}(\theta; \mathcal{X}_{t-1}) + \eta \ell(x_t, p_t, \theta). \tag{4.3}$$

This strategy has achieved strong results in tracking benchmarks [42, 88] whilst maintaining high frame-rates.

4.3.2 *Score function*

We propose a score function that is a linear combination of template and histogram scores:

$$f(x) = \gamma_{\text{tmpl}} f_{\text{tmpl}}(x) + \gamma_{\text{hist}} f_{\text{hist}}(x) \ . \tag{4.4}$$

1 The location p_1 of the object in the first frame is given.

The template score is a linear function of a K-channel feature image $\phi_x : \mathcal{T} \to \mathbb{R}^K$, obtained from x and defined on a finite grid $\mathcal{T} \subset \mathbb{Z}^2$:

$$f_{\text{tmpl}}(x; h) = \textstyle\sum_{u \in \mathcal{T}} h[u]^T \phi_x[u] \ . \tag{4.5}$$

In this, the weight vector (or *template*) h is another K-channel image.

The histogram score, instead, is computed from an M-channel feature image $\psi_x : \mathcal{H} \to \mathbb{R}^M$, obtained from x and defined on a finite grid $\mathcal{H} \subset \mathbb{Z}^2$:

$$f_{\text{hist}}(x; \beta) = g(\psi_x; \beta) \ . \tag{4.6}$$

Unlike the template score, the histogram score is invariant to spatial permutations of its feature image, such that $g(\psi) = g(\Pi\psi)$ for any permutation matrix Π. We adopt a linear function of the (vector-valued) average feature pixel

$$g(\psi; \beta) = \beta^T \left(\tfrac{1}{|\mathcal{H}|} \textstyle\sum_{u \in \mathcal{H}} \psi[u] \right) \ , \tag{4.7}$$

which can also be interpreted as the average of a scalar-valued score image $\zeta_{(\beta,\psi)}[u] = \beta^T \psi[u]$

$$g(\psi; \beta) = \tfrac{1}{|\mathcal{H}|} \textstyle\sum_{u \in \mathcal{H}} \zeta_{(\beta,\psi)}[u] \ . \tag{4.8}$$

To enable efficient evaluation of the score function in dense sliding-window search, it is important that both feature transforms commute with translation, i.e. $\phi_{T(x)} = T(\phi_x)$. Not only does this mean that overlapping windows can share feature computation, but also that the template score can be computed using fast routines for convolution, and that the histogram score can be obtained using a single integral image. Further acceleration is possible if the histogram weight vector β or feature pixels $\psi[u]$ are sparse.

It may at first seem counter-intuitive to consider f_{hist} distinct from f_{tmpl}, when it is, in fact, a special case of f_{tmpl} with $h[u] = \beta$ for all u. However, a

uniform template such as this would not be learnt from circular shifts, since the score that is obtained using a uniform template is *invariant* to circular shifts. The histogram score may thus be understood to capture an aspect of the object appearance that is lost when considering circular shifts.

To retain the speed and efficacy of the Correlation Filter without ignoring the information that can be captured by a permutation-invariant histogram score, we propose to learn our model by solving two independent ridge-regression[2] problems:

$$\begin{aligned} h_t &= \arg\min_h \left\{ \widetilde{L}_{\text{tmpl}}(h; \mathcal{X}_t) + \tfrac{1}{2}\lambda_{\text{tmpl}}\|h\|^2 \right\} \\ \beta_t &= \arg\min_\beta \left\{ \widetilde{L}_{\text{hist}}(\beta; \mathcal{X}_t) + \tfrac{1}{2}\lambda_{\text{hist}}\|\beta\|^2 \right\} \end{aligned} \tag{4.9}$$

The parameters h can be obtained quickly using the Correlation Filter formulation. While the dimension of β may be less than that of h, it may still be more expensive to solve for, since it cannot be learnt with circular shifts and therefore requires the inversion of a general matrix rather than a circulant matrix. The fast optimisation of the parameters β will be covered later in this section.

Finally, we take a convex combination of the two scores, setting $\gamma_{\text{tmpl}} = 1 - \alpha$ and $\gamma_{\text{hist}} = \alpha$ in eq. 4.4. We hope that, since the parameters of both score functions will be optimised to assign a score of 1 to the object and 0 to other windows, the magnitudes of the scores will be compatible, making a linear combination effective. Figure 4.2 is a visual representation of the overall learning and evaluation procedure.

2 Least-square with L2 regularization.

4.3.3 *Learning the template score*

Under a least-squares Correlation Filter formulation [216], the per-image loss is

$$\ell_{\text{tmpl}}(x, p, h) = \left\| \sum_{k=1}^{K} h^k \star \phi^k - y \right\|^2 \tag{4.10}$$

where h^k refers to channel k of multi-channel image h, ϕ is short for $\phi_{T(x,p)}$, y is the desired response (typically a Gaussian function with maximum value 1 at the origin), and $\star$ denotes periodic cross-correlation. This corresponds to linear regression from the circular shift of ϕ by δ pixels to the value $y[\delta]$ with a quadratic loss. Using $\hat{x}$ to denote the Discrete Fourier Transform of x, the minimiser of the regularised objective $\ell_{\text{tmpl}}(x, p, h) + \lambda \|h\|^2$ is obtained [104, 216]

$$\hat{h}[u] = (\hat{s}[u] + \lambda I)^{-1} \hat{r}[u] \tag{4.11}$$

for all $u \in \mathcal{T}$, where $\hat{s}[u]$ is a $K \times K$ matrix with elements $\hat{s}^{ij}[u]$ and $\hat{r}[u]$ is a K-dimensional vector with elements $\hat{r}^i[u]$. Treating s^{ij} and r^i as signals, these are defined

$$s^{ij} = \phi^j \star \phi^i \ , \qquad\qquad r^i = y \star \phi^i \tag{4.12}$$

or, in the Fourier domain, using * to denote conjugation and $\odot$ for element-wise multiplication,

$$\hat{s}^{ij} = (\hat{\phi}^j)^* \odot \hat{\phi}^i \ , \qquad\qquad \hat{r}^i = (\hat{y})^* \odot \hat{\phi}^i \ . \tag{4.13}$$

In practice, Hann windowing is applied to the signals to minimise boundary effects during learning. Instead of computing eq. 4.11, we use the formula-

tion in the dual of the kernelised Correlation Filter (with a linear kernel) [89, Chapter 4].

$$\hat{h}[u] = 1/(\hat{d}[u] + \lambda) \cdot \hat{r}[u] \ . \tag{4.14}$$

where $\hat{d}[u] = \mathrm{tr}(\hat{s}[u])$ or

$$\hat{d} = \sum_{i=1}^{K} (\hat{\phi}^i)^* \odot \hat{\phi}^i \tag{4.15}$$

This enables the algorithm to remain fast with a significant number of feature channels. The online version update is

$$\begin{aligned} \hat{d}_t &= (1 - \eta_{\text{tmpl}})\hat{d}_{t-1} + \eta_{\text{tmpl}}\hat{d}'_t \\ \hat{r}_t &= (1 - \eta_{\text{tmpl}})\hat{r}_{t-1} + \eta_{\text{tmpl}}\hat{r}'_t \end{aligned} \tag{4.16}$$

where $\hat{d}'$ and $\hat{r}'$ are obtained according to eq. 4.15 and eq. 4.13 respectively.

4.3.4 *Learning the histogram score*

Ideally, the histogram score should be learnt from a set of examples taken from each image, including the correct position as a positive example. Let $\mathcal{W}$ denote a set of pairs (q, y) of rectangular windows q and their corresponding regression target $y \in \mathbb{R}$, including the positive example $(p, 1)$. The per-image loss is then $\ell_{\text{hist}}(x, p, \beta) =$

$$\sum_{(q,y)\in\mathcal{W}} \left(\beta^T \left[\sum_{u\in\mathcal{H}} \psi_{T(x,q)}[u]\right] - y\right)^2. \tag{4.17}$$

For an M-channel feature transform ψ, the solution is obtained by solving an $M \times M$ system of equations, which requires $O(M^2)$ memory and $O(M^3)$ time. If the number of features is large, this is infeasible.

We instead propose features of the special form $\psi[u] = e_{k[u]}$, where e_i is a vector that is one at index i and zero everywhere else. Then, the one-sparse inner product is simply a lookup $\beta^T\psi[u] = \beta^{k[u]}$. The type of features that we consider are quantised RGB colours, although a suitable alternative would be Local Binary Patterns. Recall from eq. 4.8 that the histogram score can be considered an average vote. For efficiency, we therefore propose to apply linear regression to each feature pixel independently, over object and background regions $\mathcal{O}$ and $\mathcal{B} \subset \mathbb{Z}^2$ using the per-image objective $\ell_{\text{hist}}(x, p, \beta) =$

$$\frac{1}{|\mathcal{O}|}\sum_{u\in\mathcal{O}}\left(\beta^T\psi[u]-1\right)^2 + \frac{1}{|\mathcal{B}|}\sum_{u\in\mathcal{B}}\left(\beta^T\psi[u]\right)^2 \tag{4.18}$$

where ψ is short-hand for $\psi_{T(x,p)}$. Introducing the one-hot assumption, the objective decomposes into independent terms per feature dimension $\ell_{\text{hist}}(x, p, \beta) =$

$$\sum_{j=1}^{M}\left[\frac{N^j(\mathcal{O})}{|\mathcal{O}|}\cdot(\beta^j-1)^2 + \frac{N^j(\mathcal{B})}{|\mathcal{B}|}\cdot(\beta^j)^2\right] \tag{4.19}$$

where $N^j(\mathcal{A}) = |\{u \in \mathcal{A} : k[u] = j\}|$ is the number of pixels in the region $\mathcal{A}$ of $\phi_{T(x,p)}$ for which feature j is non-zero $k[u] = j$. The solution of the associated ridge-regression problem is

$$\beta_t^j = \frac{\rho^j(\mathcal{O})}{\rho^j(\mathcal{O})+\rho^j(\mathcal{B})+\lambda} \tag{4.20}$$

for each feature dimension $j = 1, \ldots, M$, where $\rho^j(\mathcal{A}) = N^j(\mathcal{A})/|\mathcal{A}|$ is the proportion of pixels in a region for which feature j is non-zero. This expression has previously been used under probabilistic motivation [21, 171]. In the online version, the model parameters are updated

$$\begin{aligned}\rho_t(\mathcal{O}) &= (1-\eta_{\text{hist}})\rho_{t-1}(\mathcal{O}) + \eta_{\text{hist}}\rho'_t(\mathcal{O})\\ \rho_t(\mathcal{B}) &= (1-\eta_{\text{hist}})\rho_{t-1}(\mathcal{B}) + \eta_{\text{hist}}\rho'_t(\mathcal{B})\end{aligned} \tag{4.21}$$

where $\rho_t(\mathcal{A})$ is the vector of $\rho_t^j(\mathcal{A})$ for $j = 1, \ldots, M$.

4.3.5 *Search strategy*

When searching for the target's position in a new frame, we consider rectangular windows that vary in translation/scale but not aspect ratio/orientation. Rather than search jointly in translation/scale, we search first in translation and subsequently in scale. We follow Danelljan *et al.* [42] and learn a distinct, multi-scale template for scale search using a 1D Correlation Filter. This model's parameters are updated using the same scheme as the template learnt for translation. The histogram score is not suited to scale search because it will often prefer to shrink the target to find a window that is more purely foreground.

For both translation and scale, we search only in a region around the previous location. We also follow prior works that adopt Correlation Filters for tracking [42, 88] in using a Hann window, which can be considered an implicit motion model.

The size of the translation template is normalised to have a fixed area. This parameter can be tuned to trade tracking quality for speed, as shown in the following section.

4.4 EXPERIMENTS

We compare *Staple* to competing methods on three popular benchmarks, VOT-2014 [109], VOT-2015 [110] and OTB-2013 [232]. To achieve an up-to-date comparison, we report the results of several recent trackers in addition to the baselines that are part of each benchmark, using the authors' own results.

Hyperparameter	Value
Learning rate (template) η_{tmpl}	0.01
Learning rate (histogram) η_{hist}	0.04
Colour features	RGB
Num. bins per channel	32
Merge factor α	0.3
Fixed target area	22500
HOG cell size	4×4

Table 4.1: The fixed hyperparameters we use for all our experiments.

Hyperparameters. In Table 4.1, we report the values of the most critical hyperparameters we use, chosen so that to they maximise the performance on VOT-2015. We then keep them fixed across all the experiments. The reason we chose VOT-2015 is that, at the time of submission of the original paper [18], the results of the VOT-2015 challenge had not been disclosed yet. We then included the experiments on VOT-2015 post-acceptance, following the suggestion of one of the reviewers. Note that this is still in accord with convention, since in 2015 the VOT challenge has not included a validation set.

4.4.1 *Results on VOT-2014 and VOT-2015*

The benchmark. VOT-2014 [109] and VOT-2015 [110] compare competing trackers respectively on 25 and 60 sequences chosen from a pool of 394 [198] to represent several challenging situations: camera motion, occlusion, illumination, size and motion change. Two performance measures are used. The *accuracy* of a tracker on a sequence is expressed as the average per-frame overlap between its predicted bounding box r_t and the ground truth r_{GT} using the intersection-over-union (IOU) criterion $S_t = \frac{|r_t \cap r_{\text{GT}}|}{|r_t \cup r_{\text{GT}}|}$. The *robustness* of a tracker, instead, is its number of failures over the sequence, with a failure determined to have occurred when S_t becomes zero. Importantly, since the focus

Tracker	Year	Venue	Accuracy ↑	Failures ↓	Rank ↓
Staple	-	-	**0.644**	9.38	**4.37**
DATs [171]	2015	CVPR	0.580	13.17	5.39
PLT_13 [109]	2013	VOT	0.523	1.66	5.41
DGT [30]	2014	TIP	0.534	13.78	5.66
SRDCF [45]	2015	ICCV	0.600	15.90	5.99
DMA [235]	2015	CVPR	0.476	**0.72**	6.00
PLT_14 [109]	2014	VOT	0.537	3.41	6.03
KCF [88]	2015	PAMI	0.613	19.79	6.58
DSST [42]	2014	BMVC	0.607	16.90	6.59
SAMF [128]	2014	ECCVw	0.603	19.23	6.79
DAT [171]	2015	CVPR	0.519	15.87	7.95
PixelTrack [55]	2013	ICCV	0.420	22.58	11.31

Table 4.2: Ranked results for VOT-2014 (25 sequences). Top entries for accuracy, number of failures and overall rank are reported in bold.

of the benchmark is on *short-term* tracking, a tracker that fails (i.e. $r_t \cap r_{\text{GT}} = 0$) is automatically reinitialised to the ground truth five frames after the failure.

Given the nature of the two performance measures, it is crucial to consider them jointly. Considering either in isolation is uninformative, since a tracker that fails frequently is going to be re-initialised more often and likely achieve higher accuracy, while zero failures can always be achieved by reporting that the object occupies the entire video frame.

Results. To produce Table 4.2, we used the most recent version of the VOT toolkit available at submission time (commit `d3b2b1d`). From VOT-2014 [113] we only include the top performers: DSST [42], SAMF [128], KCF [88], DGT [30], PLT_14 and PLT_13. The table reports the average accuracy and number of failures for each tracker, together with an overall ranking devised from both. Surprisingly, our simple method significantly outperforms all VOT-2014 entries, together with many recent trackers published after the challenge. In particular, it surpasses the Correlation Filter-based DSST [42], SAMF [128] and KCF [88], the colour-based PixelTrack [55], DATs [171] and DGT [30], and also more complex and far slower methods like DMA [235] and SRDCF [45],

Tracker	Year	Venue	Accuracy ↑	Failures ↓	Rank ↓
MDNet	2015	ICCV	**0.583**	**0.69**	**14.31**
DeepSRDCF	2015	VOT	0.528	1.05	19.16
SRDCF	2015	ICCV	0.521	1.24	21.01
Staple	-	-	0.533	1.39	21.64
SO-DLT	2015	arXiv	0.535	1.78	22.71
NSAMF	2015	VOT	0.490	1.29	22.93
EBT	2015	arXiv	0.453	1.02	23.01
sPST	2015	ICCV	0.508	1.48	23.04
RAJSSC	2015	VOT	0.518	1.63	23.53
SC-EBT	2015	ICML	0.523	1.86	23.70

Table 4.3: Ranked results for VOT-2015 (60 sequences). Top entries for accuracy, number of failures and overall rank are reported in bold.

which operate below 10 FPS. It is interesting to observe how *Staple* performs in comparison to the second-best correlation and colour trackers, SRDCF and DATs: it achieves a (relative) 7% improvement in accuracy and 41% improvement in number of failures over SRDCF, and an 11% improvement in accuracy and 13% improvement in number of failures over DATs. Considering the two metrics individually, *Staple* is by far the best method in terms of accuracy and the fourth for number of failures, after DMA, PLT_13 and PLT_14. However, all these trackers perform poorly in terms of accuracy.

We also compare *Staple* against the 62 trackers from the VOT-2015 competition, using the same commit of the VOT toolkit as before. Table 4.3 shows only the top ten entries and confirms the competitiveness of our method, which ranks 4th. Two of the three methods that outperform *Staple* rely on CNN features, and *all* run significantly slower: MDNet claims 1 fps, DeepSRDCF < 1 fps and SRDCF 5 fps. Furthermore, *Staple* is by far the fastest among the top ten, as it can be verified in the VOT-2015 challenge report [110].

4.4.2 *Results on OTB-2013*

The benchmark. As VOT, the idea of OTB [232] is to evaluate trackers for both accuracy and robustness to failure. Again, prediction accuracy is measured as IOU between the tracker's estimate and the ground truth. A success is detected when this value is above a certain threshold t_o. In order not to set an arbitrary value for this threshold, the area under the curve of success rates at different values of t_o is used as final score. The OTB benchmark offers three different evaluation strategies.

- **OPE** (one pass evaluation) simply runs the tracker once per video, initialising it at the very first frame.
- **TRE** (temporal robustness evaluation) augments the number of tested sequences by initialising the tracker at 20 different frames for each video.
- **SRE** (spatial robustness evaluation) initialises the tracker at the first frame of each video with a randomly perturbed bounding box. This is repeated 5 times per video.

Note that, differently from VOT, the tracker is not re-initialised after failures.

Results. Our results for OTB have been obtained using the same code and parameters used for VOT-2014 and VOT-2015. The only difference is that we are constrained to use one-dimensional histograms for the few grayscale sequences present in the benchmark. Figure 4.3 reports the results of OPE, TRE and SRE. *Staple* performs significantly better than all the methods reported in [232], with an average relative improvement of 23% with respect to the best tracker evaluated in the original benchmark (Struck [83]). Moreover, our method also outperforms recent trackers published after the benchmark, such as MEEM [244], DSST [42], TGPR [70], EBT [226] and also trackers that make use of deep convolutional neural networks like CNN-SVM [94] and SO-

DLT [227], while running at a significantly higher frame rate. The only comparable method in terms of frame-rate is ACT [43], which however performs substantially worse in all the evaluations. Since ACT learns a colour template using Correlation Filters, this result shows that the improvement that *Staple* achieves by combining template and histogram scores cannot be attributed solely to the introduction of colour. On OTB, the only tracker performing better than *Staple* is the very recent SRDCF [45]. However, it performs significantly worse on VOT-2014 and has a reported speed of only 5 FPS, which severely limits its applicability.

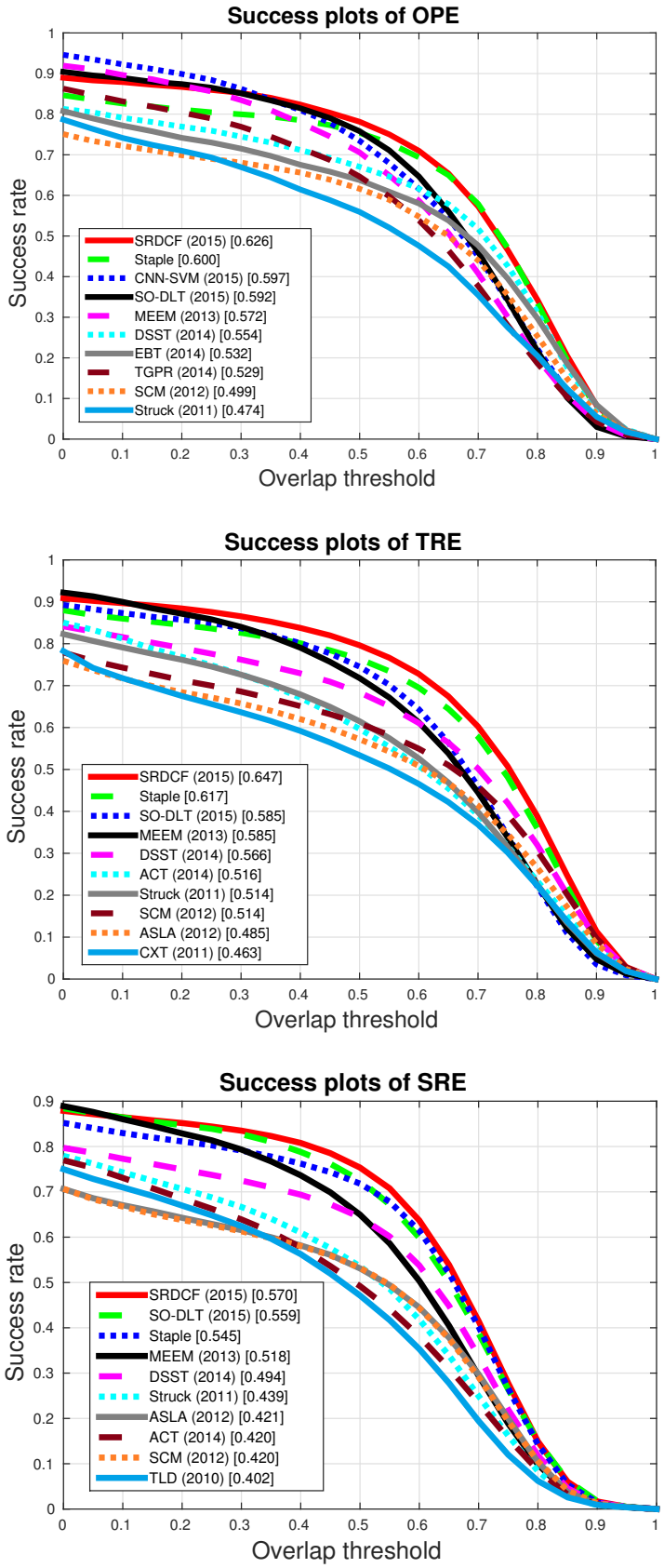


Figure 4.3: OTB-2013 [232] success plots for OPE (one pass evaluation), TRE (temporal robustness evaluation) and SRE (spatial robustness evaluation).

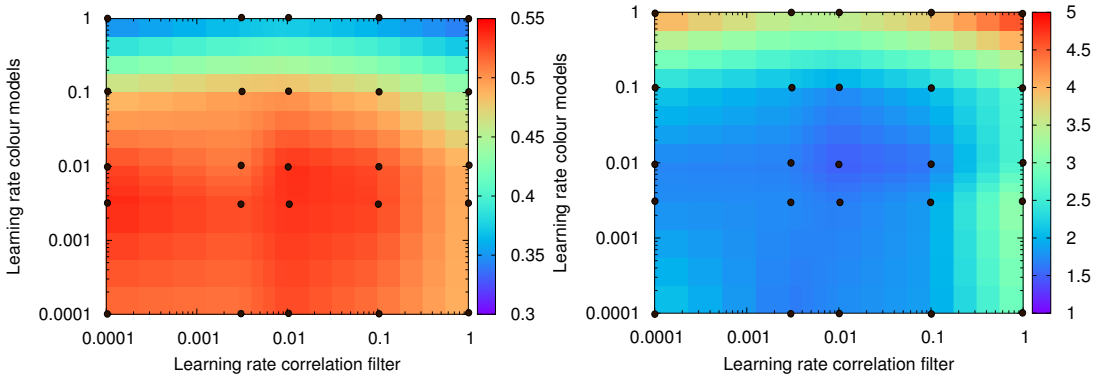


Figure 4.4: Accuracy and number of failures in relation to the learning rates η_{tmpl} and η_{hist}. Black points were obtained experimentally, others were interpolated.

4.4.3 *Further analysis*

This section empirically illustrates the impact of the most important hyperparameters on the VOT-2015 dataset.

Learning rate influence. The learning rates η_{tmpl} and η_{hist} (eq. 4.16 and 4.21) determine the rate at which to replace old evidence from earlier frames with new evidence from the current frame. The lower the learning rate, the higher the relevance given to model instances learnt in earlier frames. The heatmaps of Figure 4.4 illustrate how the best performance is achieved at around 0.01 for both η_{tmpl} and η_{hist}.

Merge factor influence. In Figure 4.5, we show how the accuracy of *Staple* is significantly influenced by the choice of the merge factor α that regulates γ_{tmpl} and γ_{hist} in eq. 4.4: the best performance is achieved around $\alpha = 0.3$. The robustness follows a similar trend. Figure 4.5 also shows that the strategy of merging the dense responses of the two ridge regression problems achieves significantly better performance than a mere interpolation of the final estimates would, suggesting that choosing models with compatible (and complementary) dense responses is a winning choice.

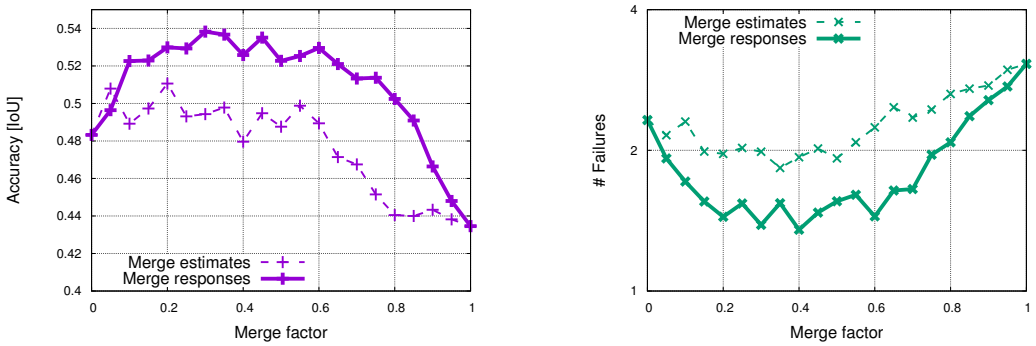


Figure 4.5: Accuracy and number of failures vs. merge factor α.

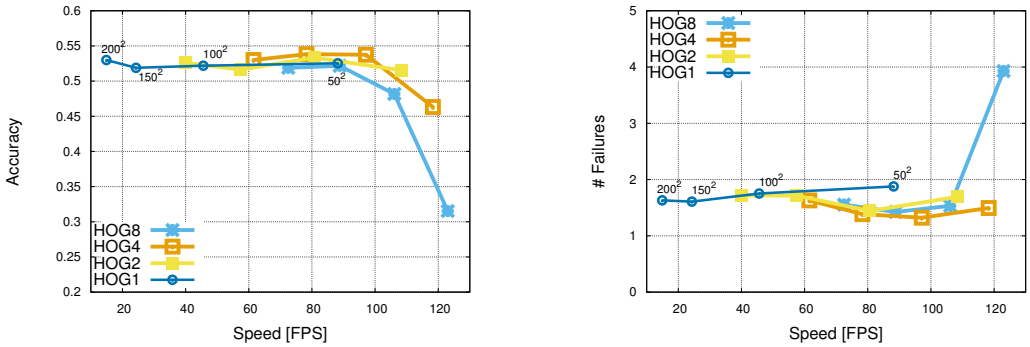


Figure 4.6: Accuracy and number of failures in relation to speed for HOG cells of size 1×1, 2×2, 4×4 and 8×8 and different fixed areas (50^2, 100^2, 150^2 and 200^2).

Speed. Our MATLAB prototype runs at approximately 80 fps on a machine equipped with an Intel Core i7@4.0GHz. However, it is possible to achieve higher frame rates with a relatively small loss in terms of performance, by adjusting the size of the patch from which the models are computed. For example (Figure 4.6), using HOG cells of size 2×2 and a fixed area of 50^2 causes only a small increase in the number of failures, yet boosts the speed beyond 100 frames per second. One possible reason is that increasing/reducing HOG cell size and fixed area size causes effects in two opposite directions. Smaller areas attain coarser estimations but on the other hand they are less sensitive to deformation.

By learning their model from circular shifts of positive examples, Correlation Filter-based trackers fail to learn a component that is invariant to permutations. This makes them inherently sensitive to shape deformation. Therefore, we proposed a simple combination of template and histogram scores that are learnt independently to preserve real-time operation. The resulting method, *Staple*, outperforms significantly more complex trackers in several benchmarks. Given its speed and simplicity, our tracker is a sensible choice for applications that require computational effort themselves, and in which robustness to colour, illumination and shape changes is paramount.

However, we believe that the potential of *Staple* is inherently limited by its design. While it is notable that the combination of colour statistics and HOG-based templates can address the broad range of scenarios covered by standard benchmarks, manually designing and combining features to address different scenarios is cumbersome. *Staple*, like most standard trackers, is constrained to learn exclusively using data sampled from the video at hand, thus neglecting labelled video datasets from which it would be possible to distil useful knowledge. In the next chapter, we attempt to address this issue by formulating the problem of object tracking in a significantly different way.

5

FULLY-CONVOLUTIONAL SIAMESE NETWORKS FOR OBJECT TRACKING

5.1 INTRODUCTION

For several years, the most successful paradigm for arbitrary object tracking has been to learn a model of the object's appearance in an online fashion using examples extracted from the video itself [198]. This owes in large part to the demonstrated ability of methods like TLD [102], Struck [83] and KCF [88]. Considering that a user may request to track any arbitrary object, it is impossible for a standard tracker to have gathered the data necessary to train a specific detector beforehand. However, by using data derived exclusively from the current video, only comparatively simple models can be learnt.

While other problems in computer vision have seen an increasingly pervasive adoption of deep convolutional networks (CNNs) trained from large supervised datasets, the scarcity of supervised data and the constraint of real-time operation have prevented the straightforward application of deep learning within this paradigm of learning a detector per video. Several recent works have aimed to overcome this limitation using a CNN pre-trained for a different but related task. These approaches either apply "shallow" methods (e.g. the Correlation Filter) using the network's internal representation as features [44, 136] or perform SGD to fine-tune multiple layers of the network [156, 224, 227]. While the use of shallow methods does not take full

advantage of the benefits of end-to-end learning, methods that apply SGD during tracking to achieve state-of-the-art results have not been able to operate in real-time.

In this chapter, we advocate an alternative approach in which a CNN is trained to address a more general *similarity learning* problem in an initial offline phase. Then, this function is simply evaluated online during tracking, without being adapted. The key contribution is to demonstrate that this novel approach achieves very competitive performance in modern tracking benchmarks at a speed that exceeds the frame-rate requirement. Specifically, we train a Siamese network to locate an *exemplar* image within a larger *search* image. We propose a Siamese architecture that is *fully-convolutional* with respect to the search image: dense and efficient sliding-window evaluation is achieved with a layer that computes the cross-correlation of its two inputs.

We posit that the similarity learning approach has gone relatively neglected because the tracking community did not have access to vast labelled datasets. In fact, until recently, the available datasets only totalled a few hundred annotated videos [113, 150, 198, 232]. However, we believe that the emergence of the ILSVRC dataset for object detection in video [184] (henceforth VID) makes it possible to train such a model. Furthermore, the fairness of training and testing deep models for tracking using videos from the same domain is a point of controversy, as it has been recently prohibited by the VOT committee. We show that our model generalises from the ImageNet Video domain to the ALOV/OTB/VOT [113, 198, 232] domain, enabling the videos of tracking benchmarks to be reserved for testing purposes.

Several recent works have sought to train Recurrent Neural Networks (RNNs) for the problem of object tracking. Gan et al. [69] suggested training an RNN to predict the absolute position of the target in each frame. Similarly, Kahou et al. [100] proposed an RNN for tracking trained with an attention mechanism. While these methods certainly represent a promising avenue for future research, they have not yet demonstrated competitive results on modern benchmarks. We remark that an interesting parallel can be drawn between this approach and ours, by interpreting a Siamese network as an unrolled RNN that is trained and evaluated on sequences of length two. Therefore, Siamese networks could serve as strong initialisation for a recurrent model.

Denil et al. [49] introduced an algorithm that tracks objects with a particle filter that uses a learnt distance metric to compare the current appearance to that of the first frame. However, their distance metric is vastly different from ours. Instead of comparing images of the entire object, they compute distances between fixations (foveated glimpses of small regions within the object's bounding box). To learn a distance metric, they train a Restricted Boltzmann Machine (RBM) and then use the Euclidean distance between hidden activations for two fixations. Although RBMs are unsupervised, they suggest training the RBM on random fixations within centred images of the object to detect. This must either be performed online or in an offline phase with knowledge of the object to track. While tracking an object, they learn a stochastic policy for choosing fixations which is specific to that object. Besides synthetic sequences of MNIST digits, this method has only been demonstrated qualitatively on problems of face and person tracking.

While it is infeasible to train a deep CNN from scratch for each new video, several works have investigated the feasibility of adapting, at test-time, a net-

work that has been pre-trained offline. SO-DLT [227] and MDNet [156] both train a convolutional network for a similar detection task in an offline phase, then at test-time use SGD to learn a detector with examples extracted from the video itself. These methods cannot operate at frame-rate due to the computational burden of evaluating forward and backward passes on many examples. An alternative way to leverage CNNs for tracking is to apply traditional shallow methods using the internal representation of a pre-trained convolutional network as features. While trackers in this style such as DeepSRDCF [44], Ma et al. [136] and FCNT [224] have achieved strong results, they have been unable to achieve frame-rate operation due to the high dimension of the CNN representation.

Concurrently with our work, some other authors have also proposed using CNNs for object tracking by learning a function of pairs of images. In GOTURN [86], a CNN is trained to regress directly from two images to the location in the second image of the object shown in the first image. Predicting a rectangle instead of a position has the advantage that changes in scale and aspect ratio can be handled without resorting to exhaustive evaluation. However, a disadvantage of their approach is that it does not possess intrinsic invariance to translations of the second image. This means that the network must be shown examples in all positions, which requires considerable dataset augmentation. YCNN [34] trains a network that maps an exemplar and a larger search region to a response map. However, their method too lacks invariance to translations of the second image, since the final layers are fully connected. Similarly to GOTURN, this is inefficient because the training set must represent all translations of all objects. Finally, with SINT [206], Tao et al. propose to train a Siamese network to identify candidate image locations that match the initial object appearance. In contrast to our approach, they do not adopt an architecture which is fully convolutional with respect to the

search image. Instead, at test-time, they sample bounding boxes uniformly on circles of varying radius as in Struck [83]. Moreover, they incorporate optical flow and bounding-box regression to improve the results. To improve the computational speed of their system, they employ Region of Interest (RoI) pooling [179] to efficiently examine many overlapping sub-windows. Despite this optimisation, at 2 frames per second, the overall system is still far from being real-time.

All of the competitive methods above that train on video sequences (MDNet [156], SINT [206], GOTURN [86]), use training data belonging to the same ALOV/OTB/VOT domain used by the benchmarks. This practice has been forbidden in the VOT challenge due to concerns about overfitting to the scenes and objects in the benchmark. Thus, an important contribution of our work is to demonstrate that a CNN can be trained for effective object tracking without using videos from the same distribution as the test set.

5.3 METHOD

Learning to track arbitrary objects can be addressed using similarity learning. We propose to use a dataset of videos with labelled object trajectories to learn a function $f(z, x)$ that compares an exemplar image z to a candidate image x of the same size and returns a high score if the two images depict the same object and a low score otherwise. To find the position of the object in a new image, we can then exhaustively test all possible locations and choose the candidate with the maximum similarity to the appearance of the object. In experiments, we will use the initial appearance of the object as the *exemplar*. Avoiding any update to the object's appearance has the advantage of keeping the overall tracking system very simple, allowing us to understand the strength of the learnt function better. Given their widespread success in

computer vision [53, 117, 164, 175], we will use a CNN as the function f. Similarity learning with deep conv-nets is typically addressed using Siamese architectures [28, 204, 241], which apply an identical transformation φ to both inputs and then combine their representations using a similarity metric g according to $f(z, x) = g(\varphi(z), \varphi(x))$. Deep Siamese conv-nets have previously been applied to tasks such as face verification [164, 194, 204], keypoint descriptor learning [197, 241] and few-shot classification [107, 199, 222].

5.3.1 *Fully-convolutional Siamese architecture*

We propose a Siamese architecture which is *fully-convolutional* with respect to the search image x, which contains multiple candidates. We say that a function is fully-convolutional if it commutes with translation. To give a more precise definition, introducing L_τ to denote the translation operator $(L_\tau x)[u] = x[u - \tau]$, a function h is fully-convolutional with integer stride k if

$$h(L_{k\tau} x) = L_\tau h(x) \tag{5.1}$$

for any translation τ.

The advantage of a fully-convolutional network is that, instead of a candidate image of the same size, we can provide as input to the network a much larger *search* image and it will compute the similarity at all translated subwindows on a dense grid in a single evaluation. To achieve this, we use a convolutional embedding function φ and combine the resulting feature maps using a cross-correlation layer

$$f(z, x) = \varphi(z) * \varphi(x) + b\,\mathbb{1}, \tag{5.2}$$

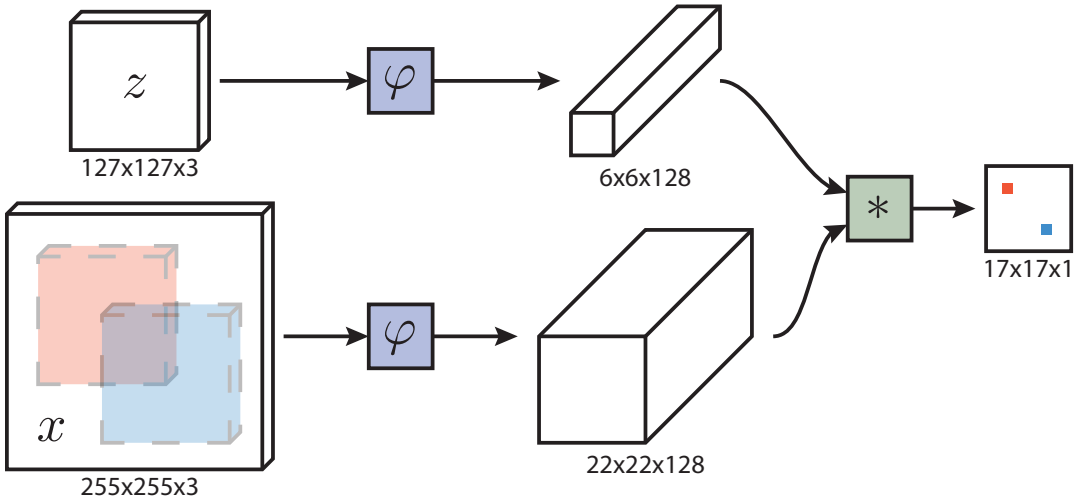


Figure 5.1: Fully-convolutional Siamese architecture. Our architecture is fully-convolutional with respect to the search image x. The output is a scalar-valued score map whose dimension depends on the size of the search image. This enables the similarity function to be computed for *all* translated sub-windows within the search image in one evaluation. In this example, the red and blue pixels in the score map contain the similarities for the corresponding sub-windows.

where $b\mathbb{1}$ denotes a signal which takes value $b \in \mathbb{R}$ in every location. The output of this network is not a single score, but rather a score map defined on a finite grid $\mathcal{D} \subset \mathbb{Z}^2$, as illustrated in Figure 5.1. Note that the output of the embedding function is a feature map with spatial support as opposed to a plain vector. A similar technique has been applied in contemporary work on stereo matching [135].

During tracking, we use a search image centred at the previous position of the target. The position of the maximum score relative to the centre of the score map, multiplied by the stride of the network, gives the displacement of the target from one frame to the next. Multiple scales are searched in a single forward-pass by assembling a mini-batch of scaled images.

Combining feature maps using cross-correlation and evaluating the network once on the larger search image is equivalent to combining feature maps using the inner product and evaluating the network on each translated sub-window independently. However, the cross-correlation layer provides an

incredibly simple method to implement this operation efficiently within the framework of existing deep learning libraries. While this is undoubtedly useful during testing, it can also be exploited during training.

5.3.2 *Training with large search images*

We employ a discriminative approach, training the network on positive and negative pairs and adopting the logistic loss

$$\ell(y, v) = \log(1 + \exp(-yv)), \tag{5.3}$$

where v is the real-valued score of a single exemplar-candidate pair and $y \in \{+1, -1\}$ is its ground-truth label. We exploit the fully-convolutional nature of our network during training by using pairs that comprise an exemplar image and a larger search image. This will produce a map of scores, effectively generating many examples per pair. We define the loss of a score map to be the mean of the individual losses

$$L(y, v) = \frac{1}{|\mathcal{D}|} \sum_{u \in \mathcal{D}} \ell(y[u], v[u]) \ , \tag{5.4}$$

requiring a true label $y[u] \in \{+1, -1\}$ for each position $u \in \mathcal{D}$ in the score map. The parameters of the conv-net θ are obtained by applying Stochastic Gradient Descent (SGD) to the problem

$$\arg\min_{\theta} \mathop{\mathbb{E}}_{(z,x,y)} L(y, f(z, x; \theta)) \ . \tag{5.5}$$

Pairs are obtained from a dataset of annotated videos by extracting exemplar and search images that are centred on the target, as shown in Figure 5.2. We believe that it is effective to consider search images centred on the target

Figure 5.2: Pairs of training samples (exemplars z and corresponding search areas x) extracted from the same video. When a sub-window extends beyond the extent of the image, the missing portions are filled with the frame's mean RGB value.

because it is likely that the most challenging sub-windows, and those which have the most influence on the performance of the tracker, are those adjacent to the target. The images are extracted from two frames of a video that both contain the object and are at most T frames apart. The class of the object is ignored during training. The scale of the object within each image is normalised without corrupting the aspect ratio of the image. The elements of the score map are considered to belong to a positive example if they are within radius R of the centre (accounting for the stride k of the network):

$$y[u] = \begin{cases} +1 & \text{if } k\|u - c\| \leq R \\ -1 & \text{otherwise} \quad . \end{cases} \tag{5.6}$$

The losses of the positive and negative examples in the score map are weighted to eliminate class imbalance.

Note that, since our network is fully-convolutional, there is no risk that it learns a bias for the sub-window at the centre.

5.3.3 *Training dataset*

The 2015 edition of ImageNet Large Scale Visual Recognition Challenge [184] (ILSVRC) introduced the ImageNet Video dataset (VID) as part of the new *object detection from video* challenge. Participants are required to classify and locate objects from 30 different classes of animals and vehicles. Training and validation sets together contain almost 4500 videos, for a total of more than one million annotated frames. These numbers are particularly impressive if compared to the number of labelled sequences in VOT [110], ALOV [198] and OTB [232], which together total less than 500 videos.

We believe that this dataset should be of extreme interest to the tracking community not only for its vast size, but also because it depicts scenes and objects different to those found in the canonical tracking benchmarks. For this reason, it can safely be used to train a deep model for tracking without overfitting to the domain of videos used in these benchmarks.

5.3.4 *Dataset curation*

During training, we adopt exemplar images that are 127×127 and search images that are 255×255 pixels. Images are scaled such that the bounding box, plus an added margin for context, has a fixed area. More precisely, if the tight bounding box has size (w, h) and the context margin is p, then the scale factor s is chosen such that the area of the scaled rectangle is equal to a constant

$$s(w + 2p) \times s(h + 2p) = A \ . \tag{5.7}$$

We use the area of the exemplar images $A = 127^2$ and set the amount of context to be half of the mean dimension, i.e. $p = (w + h)/4$. Exemplar and

Layer	Support	Chan. map	Stride	Exemplar activation size
—	—	—	—	$127 \times 127 \times 3$
CONV1	11×11	96×3	2	$59 \times 59 \times 96$
POOL1	3×3	—	2	$29 \times 29 \times 96$
CONV2	5×5	256×48	1	$25 \times 25 \times 256$
POOL2	3×3	—	2	$12 \times 12 \times 256$
CONV3	3×3	384×256	1	$10 \times 10 \times 192$
CONV4	3×3	384×192	1	$8 \times 8 \times 192$
CONV5	3×3	256×192	1	$6 \times 6 \times 128$

Table 5.1: Architecture of the proposed convolutional embedding function, which is similar to the convolutional stage of the network of AlexNet [117]. The channel map property describes the number of output and input channels of each convolutional layer.

search images for every frame are extracted offline to avoid image resizing during training. For training, we use all 4417 videos of ImageNet Video, which account for more than 2 million labelled bounding boxes.

5.3.5 *Network architecture*

The architecture that we adopt for the embedding function φ resembles the convolutional stage of AlexNet [117]. The dimensions of the parameters and activations are given in Table 5.1. Max-pooling is employed after the first two convolutional layers. ReLU non-linearities follow every convolutional layer except for conv5, the final layer. During training, batch normalization [98] is inserted immediately after every linear layer. The stride of the final representation is eight. An important aspect of the design is that no padding is introduced within the network. Although this is common practice in image classification, it violates the fully-convolutional property of eq. 5.1, as it would make the embedding function φ able to learn that the boundaries of the search image correspond to a low score.

5.3.5.1 *Tracking algorithm*

Since our purpose is to prove the efficacy of our fully-convolutional Siamese network and its generalisation capability when trained on ImageNet Video, we use a simplistic algorithm to perform tracking. Unlike more sophisticated trackers, we do not update the model (e.g. [18, 45, 88]) or maintain a memory of past appearances (e.g. [83, 238]) , we do not incorporate additional cues such as optical flow (e.g. [206]) or colour histograms (e.g. [18]), and we do not refine our prediction with bounding-box regression (e.g. [156, 206]).

Despite its simplicity, this tracker achieves surprisingly good results when equipped with our offline-learnt similarity metric. Online, we do incorporate some elementary temporal constraints: we only search for the object within a region of approximately four times its previous size, and a Hann window is added to the score map to penalise large displacements. Tracking through scale space is achieved by processing several scaled versions of the search image. Any change in scale is penalised, and updates of the current scale are dampened.

5.4 EXPERIMENTS

5.4.1 *Implementation details*

Training. The parameters of the CNN are found by minimising eq. 5.5 with SGD. The initial values of the parameters follow a Gaussian distribution, scaled according to the Xavier method [85]. Training is performed over 50 epochs, each consisting of 50,000 pairs, sampled uniformly between 1 and 100 frames (Section 5.3.2). The gradients for each iteration are estimated using

mini-batches of size 8, and the learning rate is annealed (logarithmically) at each epoch from 10^{-2} to 10^{-5}.

Tracking. As mentioned earlier, the online phase is deliberately minimalistic. The embedding $\varphi(z)$ of the initial object appearance is computed only once (at the first frame of a video), and it is compared convolutionally to sub-windows of the subsequent frames. We found that updating the feature representation of the exemplar online through simple strategies, such as linear interpolation, does not gain much performance and thus we keep it fixed. We found that upsampling the score map using bicubic interpolation, from 17×17 to 272×272, results in more accurate localisation since the original map is relatively coarse. To handle scale variations, we also search for the object over five scales $1.025^{\{-2,-1,0,1,2\}}$, and update the scale by linear interpolation with a factor of 0.35 to provide damping.

On a machine equipped with a single NVIDIA Titan X and an Intel Core i7 @4.0GHz, our full online tracking pipeline operates at 55 and 85 frames per second (fps) when searching respectively over 5 or 3 scales.

5.4.2 *Results on OTB-2013*

In addition to the trackers reported by the original benchmark paper of Wu et al. [232] [1], in Figure 5.3 we compare our method against seven additional methods presented at recent major computer vision conferences and that can run in real-time: *Staple* [18], LCT [137], CCT [246], SCT4 [36], DLSSVM_NU [158], DSST [42] and KCFDP [96]. We use the names SiamFC and SiamFC_3s to refer to two slightly different variants of our method using 5 and 3 scale respectively. To reflect the nature of the sequences, for this benchmark only we train

1 For a description of the evaluation protocols, refer to Section 4.4 and Wu et al. [232].

our network with pairs of images that are grayscale with 25% chance and RGB otherwise.

Except for DLSSVM, which is a structured SVM-based method that extends Struck [83], all the other trackers are based on the Correlation Filter and often make use of ensembles. For instance, LCT [137] combines a KCF [88] tracker updated in every frame with a second Correlation Filter (updated more conservatively) for failure detection. If the target is lost, a third module based on random forests is tasked to re-detect it. Similarly, KCFDP [96] proposes to hedge KCF's lack of robustness with an object proposal module [249]. Instead, SCT4 [36] instantiates and updates multiple Correlation Filters trained on HOG features within a part-based scheme.

Surprisingly, despite its extreme simplicity and the lack of online adaptation, SiamFC improves over the state of the art for real-time tracking, performing favourably against methods based on significantly more established techniques.

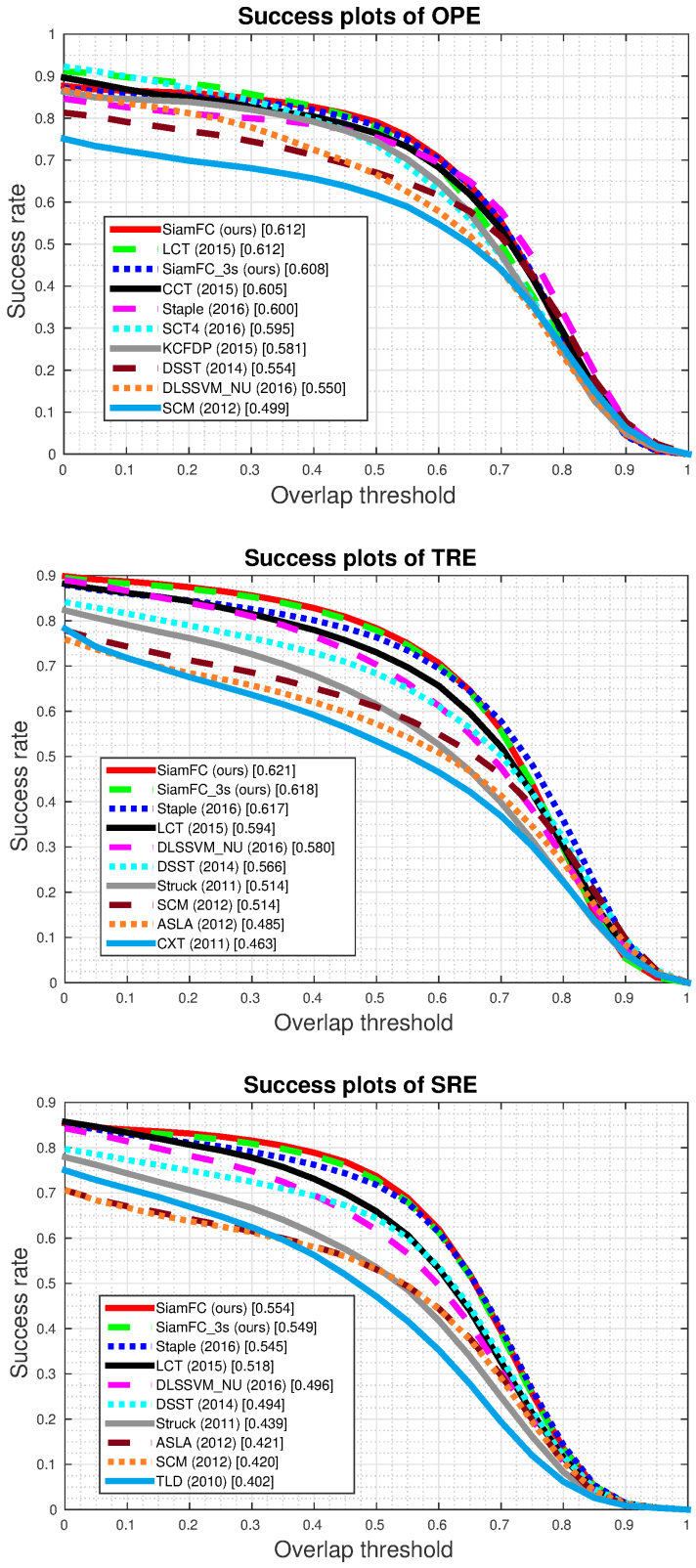


Figure 5.3: OTB-2013 [232] success plots for OPE (one pass evaluation), TRE (temporal robustness evaluation) and SRE (spatial robustness evaluation). For a description of the evaluation protocols, please refer to Section 4.4 and Wu et al. [232].

5.4.3 *Results on VOT-2017*

While our fully-convolutional Siamese approach has been first presented in our 2016 paper [19], its performance and generalisation capability have been extensively validated at a subsequent time.

Notably, we took part in the 2017 edition of the annual VOT challenge with a marginally modified version of the method described in Section 5.3. Specifically, we reduced the total stride of the CNN from 8 to 4, reduced the number of output channel from 256 to 32 and conducted a random search to establish the value of the few hyperparameters of SiamFC on a validation set of 129 video sequences (disjoint from the test set of VOT-2017). We also entered the competition with *Staple* (described in Chapter 4), deciding the hyperparameters in the same way. Further details are provided in Section 6.5.

Figure 5.4 illustrates the results of the real-time challenge of VOT-2017. Beside showing accuracy and robustness to failures on a 2D plot, the figure also reports the ranking of *all* the competing trackers by combining accuracy and number of failures in a single measure, the *expected average overlap* (EAO). As it is standard for the VOT protocol, the trackers are re-initialised five frames after a failure. Moreover, for this experiment, the videos are "fed" to the tracking algorithms by the VOT-toolkit at a typical video framerate. If a new frame is available *before* the tracker has responded with an estimate of the target's position, its previously reported position is used instead. This means that slow trackers are penalised the further their speed is from the real-time requirement (25 fps). However, for the sake of the challenge, there is no actual incentive to operate *faster* than real-time.

Notably, SiamFC and *Staple* are the second and fourth best methods (out of 51) according to the EAO plot of Figure 5.4b. Except for ASMS [223], which

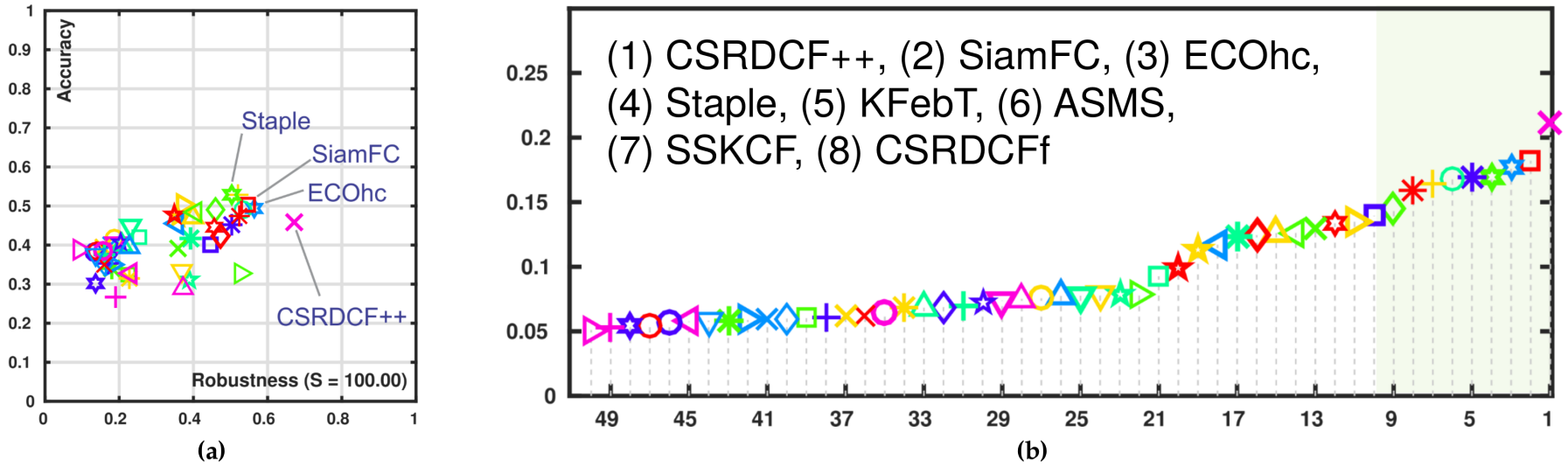


Figure 5.4: Accuracy/Robustness plot (a) and Expected Average Overlap ranking (b) for the VOT-2017 real-time challenge. For both plots, the best methods are those occupying the top-right corner. Figure from the presentation of the VOT-2017 [114] committee.

extends the Mean Shift framework to take into account scale variations, all the other top 8 trackers have the Correlation Filter at their core. CSRDCF++ and CSRDCFf are efficient implementations of the method proposed by Lukežič et al. [133], which learns the Correlation Filter from large patches with the aid of a spatial reliability map. Instead, ECOhc [47] improves in terms of both performance and efficiency the popular C-COT [46] by using a factorised convolution operator in the Correlation Filter and modelling the training data with a Gaussian Mixture Model, so that each component focuses on a different appearance. Finally, similarly to *Staple*, both KFebT [195] and SSKCF [114, 126] ensemble the Correlation Filter with independent learners based on colour histograms.

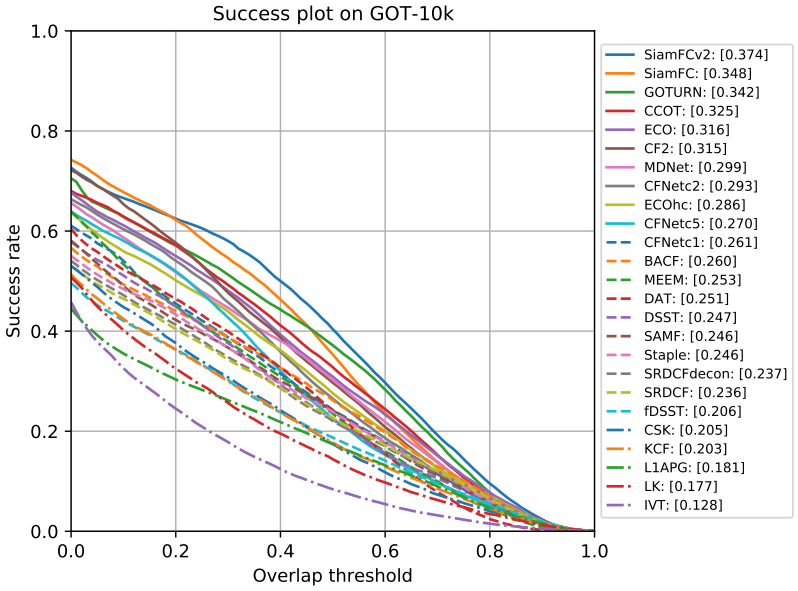
5.4.4 *Results on recent benchmarks: GOT-10k and LaSoT*

Finally, in Figure 5.5, we report precision and success plots (originally introduced by Wu et al. [232]) for two large-scale benchmarks that have been introduced after the publication of the paper on which this chapter is based [19]: GOT-10k [97] and LaSoT [58]. Surprisingly, SiamFC ranks first[2] and third, surpassing many more modern trackers that have shown outstanding results on smaller datasets [113, 233], such as ECO [47], BACF [68], TRACA [37], StructSiam [245] and PTAV [57].

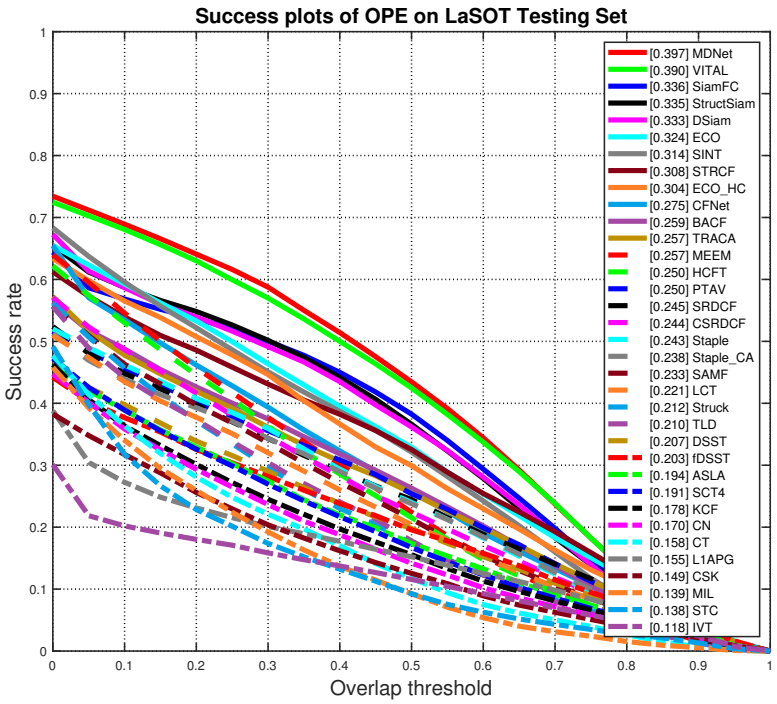
5.4.5 *Dataset size*

Table 5.2 illustrates how the size of the dataset used to train the Siamese network greatly influences the performance. The expected average overlap (mea-

2 *SiamFCv2* refers to the slightly modified version of SiamFC used as baseline in Section 5.3



(a)



(b)

Figure 5.5: Success plots [232] of two very recently introduced large scale benchmarks: (a) GOT-10k [97] and (b) LaSoT [58]. Plots from the original papers.

sured on VOT-2015) steadily improves from 0.168 to 0.274 when increasing the size of the dataset from 2% (88 videos and 60,000 annotations) to 100% (4417 videos and 2 million annotations). This finding suggests that using a

Table 5.2: Effects of using increasing portions of the ImageNet Video dataset [184] (VID) on SiamFC performance (on VOT-2015).

VID%	N.videos	N.objects	Accuracy ↑	N.failures ↓	EAO ↑
2	88	60k	0.484	183	0.168
4	177	110k	0.501	160	0.192
8	353	190k	0.484	142	0.193
16	707	330k	0.522	132	0.219
32	1413	650k	0.521	117	0.234
100	4417	2m	**0.524**	**87**	**0.274**

larger video dataset could increase the performance even further. Even if 2 million bounding boxes might seem a huge number, it should not be forgotten that they still belong to a relatively moderate number of videos, at least compared to the amount of data usually employed to train CNNs.

5.4.6 *Qualitative analysis*

Figure 5.6 provides few qualitative examples demonstrating SiamFC's efficacy in scenes exhibiting challenging appearance changes such as motion blur, non-rigid deformation, poor illumination and scale change. Despite our tracker is solely initialised at the first frame of a video and never updated, it is robust to all these scenarios.

This robustness can be attributed to the training strategy of the similarity function at the core of SiamFC, which leverages millions of positive image pairs sampled from video frames depicting the same object. Being the *search* patches sampled, in expectation, relatively far away (in time) from the *exemplar* patches, the Siamese network is exposed to scenarios in which the two inputs often depict the target with significantly different appearances (e.g. third example of Figure 5.2). This strategy can also explain why SiamFC is so delicate in situations of *confusion* (Figure 5.6, fifth row), when several instances visually similar to the target are present in the scene.

A further limitation of SiamFC is its incapacity of adapting the aspect ratio of the bounding box estimating the position of the target object, which causes a significant loss in IOU in some of the sequences (e.g. Figure 5.6, third row).

Figure 5.6: Snapshots of the simple SiamFC tracker described in Section 5.4 equipped with our proposed fully-convolutional Siamese network trained from scratch on ImageNet Video [184]. Our method does not perform any model update, so it uses only the first frame to compute $\varphi(z)$. Nonetheless, it is surprisingly robust to a number of challenging situations like motion blur (row 2), drastic change of appearance (rows 1, 3 and 4), poor illumination (row 6) and scale change (row 6). On the other hand, our method is sensitive to scenes with confusion (row 5). All sequences come from the VOT-2015 benchmark: *gymnastics1*, *car1*, *fish3*, *iceskater1*, *marching*, *singer1*. The snapshots have been taken at fixed frames (1, 50, 100 and 200) and the tracker is never re-initialised.

In this work, we departed from the traditional online learning methodology typically employed in arbitrary object tracking and showed an alternative approach that focuses on learning strong embeddings in an offline phase. We demonstrated that Siamese and fully-convolutional deep networks are particularly suited for the tracking scenario, as they have the ability to use the available data very efficiently. This is reflected both at test time, by performing efficient spatial searches, but also at training time, where every sub-window effectively represents a useful sample with little extra cost. The experiments show that deep embeddings provide a naturally rich source of features for online trackers, and enable simplistic test-time strategies to perform well. We believe that our simple approach can be used as a starting building block for more sophisticated online tracking methodologies.

One obvious design limitation of our method is that the offline-trained similarity function is evaluated *as is*. Instead, since the videos presented at test time can significantly differ from those encountered during training, it is desirable to conceive a system that is capable of adapting online. This observation is one of the primary motivations behind the contributions of the next chapter of this thesis.

6

END-TO-END REPRESENTATION LEARNING FOR THE CORRELATION FILTER

6.1 INTRODUCTION

Deep neural networks are a powerful tool for learning image representations in computer vision applications. However, training DNNs online to capture previously unseen object classes from one or a few examples is challenging. This problem emerges naturally in applications such as arbitrary object tracking. The primary challenge is the lack of prior knowledge of the target object, which can be of any class. The simplest approach is to entirely disregard this lack of prior knowledge and adapt a pre-trained deep convolutional neural network (CNN) to the target online, for example by using SGD [156, 227, 243]. Nonetheless, the minimal training data available online and the large number of parameters to learn make this a difficult learning problem. What is more, SGD is expensive for online adaptation.

A possible answer to these shortcomings is to have no online adaptation of the network at all. Recent works have focused on learning deep embeddings that can be used as universal object descriptors [19, 34, 86, 124, 206]. These methods use a Siamese CNN, trained offline to discriminate whether two image patches contain the same object or not. The idea is that a powerful embedding enables the detection (and therefore tracking) of objects via similarity, bypassing the online learning problem. However, using a fixed metric to

compare appearance prevents the learning algorithm from exploiting video-specific cues that could be helpful for discrimination.

An alternative strategy is to use instead an online learning method such as the *Correlation Filter* (CF) [118]. The CF is an efficient algorithm that learns to discriminate an image patch from the surrounding ones by solving a large ridge regression problem efficiently [25, 88]. It has proved to be highly successful in object tracking (e.g. [18, 42, 128, 137]), where its efficiency enables a tracker to adapt its internal model of the object on the fly at *every* frame. It owes its speed to a Fourier domain formulation, which allows the ridge regression problem to be solved with only a few applications of the Fast Fourier Transform (FFT) and cheap element-wise operations. Such a solution is, by design, much more efficient than an iterative solver, and still allows the discriminator to be tailored to a specific video, contrary to the embedding methods.

It is attractive, then, to try and combine the online learning efficiency of the CF with the discriminative power of CNN features trained offline. This has been done in several works (e.g. [44, 46, 47, 136, 227]), which have shown that CNNs and CFs are complementary and their combination results in improved performance. However, in these works, the CF is simply applied "on top" of pre-trained CNN features, without any integration of the two methods.

However, end-to-end training of deep architectures is generally preferable to training individual components separately. The reason is that in this manner the free parameters in all components can co-adapt and cooperate to achieve a single objective. Thus, it is natural to ask whether a CNN-CF combination can also be trained end-to-end with similar benefits. The essential step in achieving such integration is to interpret the CF as a differentiable CNN layer, so that errors can be propagated through the CF back to the CNN features. This is challenging, since the CF itself is the solution of a learning

problem. Hence, this requires to differentiate the solution of a large linear system of equations.

In this chapter, we present a closed-form expression for the derivative of the CF and we integrate it into the fully-convolutional Siamese framework presented in Chapter 5. In order to train the CNN, we adopt the meta-learning paradigm outlined in Section 3.1 and organise training in two levels. At the base level, we learn a new Correlation Filter for every pair of training images, while at the meta-level we iterate over many pairs with SGD. This allows us to learn features that are particularly well suited for the CF.

In the experimental section, we extensively investigate the empirical effect of incorporating the CF into the fully-convolutional Siamese framework. Interestingly, we find that the CF layer does not improve results for networks that are sufficiently deep. However, our method has the practical advantage of enabling ultra-lightweight networks of a few thousand parameters to achieve strong performance on multiple benchmarks while running at high framerates.

6.2 RELATED WORK

Since the seminal work of Bolme et al. [25], the Correlation Filter has enjoyed great popularity within the tracking community. Notable efforts have been devoted to its improvement, for example by mitigating the effect of periodic boundaries [45, 61, 105, 133], incorporating multi-resolution features [46, 136] and augmenting the objective with a more robust loss [181]. For the sake of simplicity, we adopt its basic formulation, which we present in Section 6.3.2.

Recently, several methods based on Siamese networks have been introduced [19, 86, 206], raising interest in the tracking community for their sim-

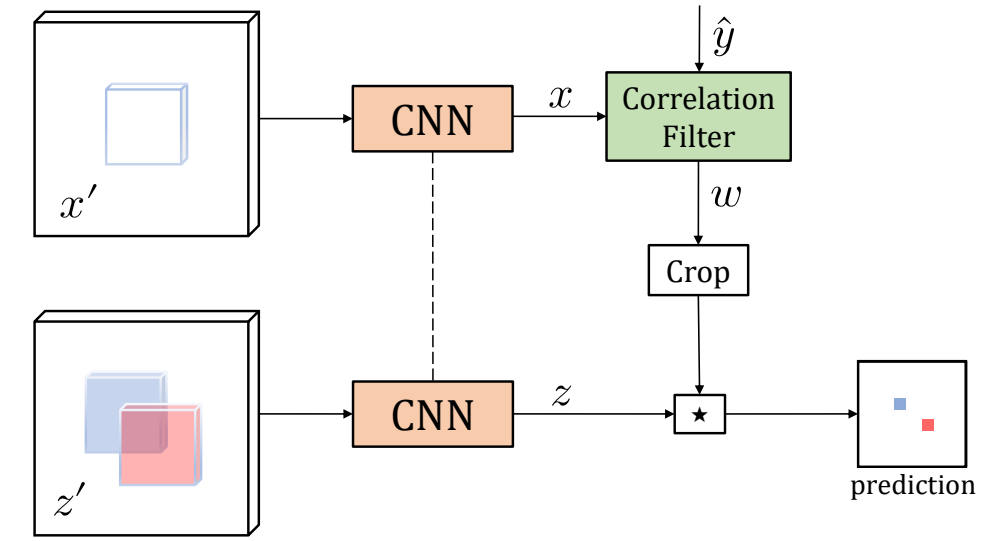
plicity and competitive performance. For our method, we prefer to build upon the fully-convolutional Siamese framework of Chapter 5, as it delivers good performance at high speed and enforces the useful prior that the appearance similarity function should commute with translation.

At its core, the Correlation Filter layer that we introduce amounts to computing the solution of a regularised deconvolution problem [48] [1]. Before it became apparent that algorithms such as SGD are sufficient for training deep networks, Zeiler et al. [242] introduced a deep architecture in which each layer solves a convolutional sparse coding problem. In contrast, our problem has a closed-form solution since the Correlation Filter employs quadratic regularisation rather than L1 regularisation.

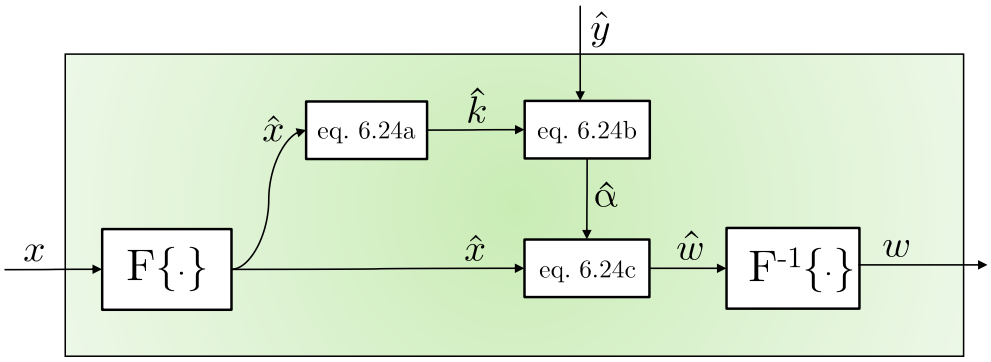
The idea of backpropagating gradients through the solution to an optimisation problem during training has been previously investigated. Ionescu et al. [99] and Murray [154] have presented backpropagation forms for the SVD and Cholesky decomposition respectively, enabling gradient descent to be applied to a network that computes the solution to either an eigenvalue problem or a system of linear equations. Our work can be understood as an efficient backpropagation procedure through the solution to a system of linear equations, where the matrix has a circulant structure.

Backpropagating through a learning algorithm invites comparison to meta-learning [220] (Section 2.3 and Chapter 3). Similarly to us, many researchers (e.g. [17, 20, 199, 222]) have also proposed to organise training in two nested "training loops", often to tackle the problem of few-shot classification [59, 123]. However, our work is the first one which considers the Correlation Filter as the base learner in a meta-learning framework.

1 Not to be confused with upsampling convolution layers that are sometimes referred to as "deconvolution layers" [131].



(a)



(b)

Figure 6.1: An overview of the proposed network architecture, **CFNet**, is illustrated on the top figure, while the bottom details the internal computational graph for the *Correlation Filter* block of the architecture. **(a)** CFNet is an asymmetric Siamese network: after applying the same convolutional feature transform to both input images x' and z', the "training image" x is used to learn a linear template w, which is then applied to search the "test image" z by cross-correlation. **(b)** The Correlation Filter (CF) layer trains a discriminative template from translations of its input by solving a system of linear equations. By backpropagating through the CF layer, we are able to optimise the image representation for the specific purpose of tracking an object with a Correlation Filter.

With the purpose of making this chapter self-contained, before introducing the Correlation Filter network and its backpropagation map, below we give a short overview of the main mathematical tools that will be used.

We use $\widehat{x} = Fx$ to denote the Discrete Fourier Transform of a variable, x^* to denote the complex conjugate, $\circ$ to denote element-wise multiplication and $\mathbb{1}$ to denote a signal of ones.

6.3.1 *Kernel linear regression*

First, consider the general linear regression problem of learning the weight vector w that best maps each of n example input vectors $x_i \in \mathbb{R}^d$ to their target $y_i \in \mathbb{R}$. The squared error can be expressed

$$\frac{1}{2n}\sum_{i=1}^{n}(x_i^T w - y_i)^2 = \frac{1}{2n}\|X^T w - y\|^2 \tag{6.1}$$

where $X \in \mathbb{R}^{d\times n}$ is a matrix whose columns are the example vectors and $y \in \mathbb{R}^n$ is a vector of the targets. Incorporating regularisation, the problem is

$$\arg\min_{w} \quad \tfrac{1}{2n}\|X^T w - y\|^2 + \tfrac{\lambda}{2}\|w\|^2 \ . \tag{6.2}$$

Kernel linear regression can be developed by writing this as a constrained optimisation problem:

$$\begin{aligned} \arg\min_{w,r} \quad & \tfrac{1}{2n}\|r\|^2 + \tfrac{\lambda}{2}\|w\|^2 \\ \text{subject to} \quad & r = X^T w - y \end{aligned} \tag{6.3}$$

and then finding a saddle point of the Lagrangian

$$L(w, r, v) = \tfrac{1}{2n}\|r\|^2 + \tfrac{\lambda}{2}\|w\|^2 + v^T(r - X^T w + y) \ . \tag{6.4}$$

The final solution can be obtained from the dual variable

$$w = \tfrac{1}{\lambda} X v \tag{6.5}$$

and the solution to the dual problem is

$$v = \tfrac{\lambda}{n} K^{-1} y \tag{6.6}$$

where $K = \frac{1}{n} X^T X + \lambda I$ is the regularised kernel matrix. It is standard to introduce a scaled dual variable $\alpha = \frac{1}{\lambda} v$ that defines w as a weighted combination of examples

$$w = X\alpha = \sum_{i=1}^{n} \alpha_i x_i \quad \text{with} \quad \alpha = \frac{1}{n} K^{-1} y \ . \tag{6.7}$$

The kernel matrix is $n \times n$ and therefore the dual solution is more efficient than the primal solution, which requires inversion of a $d \times d$ matrix, when the number of features d exceeds the number of examples n.

6.3.2 *Single-channel Correlation Filter*

Given a scalar-valued example signal x with domain $\mathcal{U}$ and corresponding target signal y, the Correlation Filter w is the scalar-valued signal

$$\arg\min_{w} \quad \frac{1}{2n}\|w \star x - y\|^2 + \frac{\lambda}{2}\|w\|^2 \tag{6.8}$$

where signals are treated as vectors in $\mathbb{R}^{\mathcal{U}}$ and the circular cross-correlation of two signals $w \star x$ is defined

$$(w \star x)[u] = \sum_{t \in \mathcal{U}} w[t] x[u + t \bmod m] \quad \forall u \in \mathcal{U} \ . \tag{6.9}$$

The solution from the previous section can then be used by defining X to be the matrix in $\mathbb{R}^{\mathcal{U} \times \mathcal{U}}$ such that $X^T w = w \star x$. It follows that the kernel matrix K belongs to $\mathbb{R}^{\mathcal{U} \times \mathcal{U}}$ and the dual variable α is a signal in $\mathbb{R}^{\mathcal{U}}$.

The key to the Correlation Filter is that the circulant structure of X enables the solution to be computed efficiently in the Fourier domain. The matrix X has elements $X[u, t] = x[u + t \bmod m]$. Since the matrix X is symmetric, the template w is obtained as cross-correlation

$$w = X\alpha = \alpha \star x \ . \tag{6.10}$$

The linear map defined by the kernel matrix K is equivalent to convolution with a signal k

$$Kz = k * z \quad \forall z \tag{6.11}$$

which is defined $k = \frac{1}{n} x \star x + \lambda \delta$, since

$$\begin{aligned} \forall z : FX^T Xz &= F((z \star x) \star x) \\ &= \widehat{z} \circ \widehat{x}^* \circ \widehat{x} = F(z * (x \star x)) \ . \end{aligned} \tag{6.12}$$

Therefore the solution is defined by the equations

$$\begin{cases} k = \frac{1}{n} x \star x + \lambda \delta \\ k * \alpha = \frac{1}{n} y \\ w = \alpha \star x \end{cases} \tag{6.13}$$

where k can be interpreted as the signal that defines a circulant linear kernel matrix, and α is a signal comprised of the Lagrange multipliers of a constrained optimisation problem that is equivalent to eq. 6.8.

6.3.3 *Adjoint of the differential*

Consider a computational graph that computes a scalar loss $\ell \in \mathbb{R}$. Within this network, consider an intermediate function that computes $y = f(x)$ where $x \in \mathcal{X} = \mathbb{R}^m$ and $y \in \mathcal{Y} = \mathbb{R}^n$. Backpropagation computes the gradient with respect to the input $\nabla_x \ell \in \mathcal{X}$ from the gradient with respect to the output $\nabla_y \ell \in \mathcal{Y}$.

The derivative $\partial f(x)/\partial x$ is a matrix in $\mathbb{R}^{n \times m}$ whose ij-th element is the partial derivative $\partial f_i(x)/\partial x_j$. This matrix relates the gradients according to

$$(\nabla_x \ell)^T = \frac{\partial \ell}{\partial x} = \frac{\partial \ell}{\partial y}\frac{\partial y}{\partial x} = (\nabla_y \ell)^T \frac{\partial f(x)}{\partial x} \tag{6.14}$$

Hence, the backpropagation map is the linear map which is the adjoint of that defined by the derivative. That is, if the derivative defines the linear map

$$J(u) = \frac{\partial f(x)}{\partial x} u, \tag{6.15}$$

then the backpropagation map is the unique linear map J^* that satisfies

$$\langle J^*(v), u \rangle = \langle v, J(u) \rangle \quad \forall u \in \mathcal{X}, v \in \mathcal{Y} \tag{6.16}$$

and the gradient with respect to the input is obtained $\nabla_x \ell = J^*(\nabla_y \ell)$. This is the core of reverse-mode differentiation [81].

We now introduce the Correlation Filter network (CFNet) and derive the expressions for evaluation and backpropagation of the Correlation Filter layer, which performs online learning during the network's forward pass.

6.4.1 *Fully-convolutional Siamese networks*

Our starting point is a network similar to that of Chapter 5, which we modify to allow the model to be interpreted as a Correlation Filter tracker. The fully-convolutional Siamese framework considers pairs (x', z') comprising a training image x' and a test image z' [2]. The image x' represents the object of interest (e.g. an image patch centred on the target object in the first video frame), while z' is typically larger and represents the search area (e.g. the next video frame).

Both inputs are processed by a CNN f_ρ with learnable parameters ρ. This yields two feature maps, which are then cross-correlated:

$$g_\rho(x', z') = f_\rho(x') \star f_\rho(z') \ . \tag{6.17}$$

Eq. 6.17 amounts to performing an exhaustive search of the pattern x' over the test image z'. The goal is for the maximum value of the response map (left-hand side of eq. 6.17) to correspond to the target location.

To this end, we train the network offline with millions of random pairs (x'_i, z'_i) taken from a collection of videos. Each example has a spatial map of labels c_i with values in $\{-1, 1\}$, with the true object location belonging to

2 Note that, for the sake of convenience, the notation here slightly differs from the one of Chapter 5, where target object and search area are denoted by z and x respectively.

the positive class and all others to the negative class. Training proceeds by minimising an element-wise logistic loss ℓ over the training set:

$$\arg\min_{\rho} \quad \sum_{i} \ell \left(g_{\rho}(x'_i, z'_i),\, c_i \right) \quad . \tag{6.18}$$

6.4.2 *Tracking algorithm*

To apply CFNet to object tracking, it is necessary to combine it with a procedure that describes the logic of the tracker. Similar to Chapter 5, we employ a very simplistic online tracking algorithm in which the network is evaluated in forward mode, once per frame.

The feature representation of the target object is compared to that of the search region, which is obtained in each new frame by extracting a window centred at the previously estimated position, with an area that is four times the size of the object. The new position of the object is taken to be the location with the highest score.

The fully-convolutional Siamese network introduced in Chapter 5 simply compares every frame to the initial appearance of the object. In contrast, similarly to modern Correlation Filter-based trackers (e.g. [42, 45, 88, 137]), we compute a new template in each frame and then combine this with the previous template in a moving average.

6.4.3 *The Correlation Filter Network*

We propose to modify the baseline Siamese network of eq. 6.17 with a differentiable Correlation Filter layer between x' and the cross-correlation operator.

The resulting architecture is illustrated in Figure 6.1. This change can be formalized as:

$$h_{\rho,s,b}(x', z') = s\,\omega\left(f_\rho(x')\right) \star f_\rho(z') + b \tag{6.19}$$

The CF block $w = \omega(x)$ computes a standard CF template w from the training feature map $x = f_\rho(x')$ by solving a ridge regression problem in the Fourier domain [88]. Its effect can be understood as crafting a discriminative template that is robust against translations. It is necessary to introduce scalar parameters s and b (scale and bias) to make the score range suitable for logistic regression. Offline training is then performed in the same way as for a Siamese network (Section 5.3.2), replacing g with h in eq. 6.18.

We found that it is important to provide the Correlation Filter with a large region of context in the training image, which is consistent with the findings of Danelljan et al. [45] and Kiani et al. [105]. To reduce the effect of circular boundaries, the feature map x is pre-multiplied by a cosine window [25] and the final template is cropped [217].

Notice that the forward pass of the architecture in Figure 6.1 corresponds exactly to the operation of a standard CF tracker [19, 42, 88, 137] with CNN features, as proposed in previous work [44, 136]. However, these earlier networks were not trained end-to-end. The novelty here is to compute the derivative of the CF template with respect to its input, so that the overall network can be trained end-to-end, thus tailoring the CNN output activations for the Correlation Filter.

6.4.4 *Backpropagating through the Correlation Filter*

We now show how to backpropagate gradients through the Correlation Filter solution efficiently and in closed form via the Fourier domain.

Recall that the optimal template w must satisfy the system of equations obtained via the Lagrangian dual (Section 6.3.2). Its solution can be computed in the Fourier domain [88],

$$\begin{cases} \widehat{k} = \frac{1}{n}(\widehat{x}^* \circ \widehat{x}) + \lambda \mathbb{1} & \text{(6.20a)} \\ \widehat{\alpha} = \frac{1}{n}\widehat{k}^{-1} \circ \widehat{y} & \text{(6.20b)} \\ \widehat{w} = \widehat{\alpha}^* \circ \widehat{x} & \text{(6.20c)} \end{cases}$$

The inverse of element-wise multiplication is element-wise scalar inversion. Notice that the operations in eq. 6.20 are more efficiently computed in the Fourier domain, since they involve element-wise operations instead of more expensive convolutions or matrix operators (eq. 6.13). Moreover, the inverse convolution problem (to find α such that $k * \alpha = \frac{1}{n}y$) is the solution to a diagonal system of equations in the Fourier domain (eq. 6.20b).

Backpropagation map (single-channel). We adopt the notation that if $x \in \mathcal{X} = \mathbb{R}^n$ is a variable in a computational graph that computes a final scalar loss $\ell \in \mathbb{R}$, then $\nabla_x \ell \in \mathcal{X}$ denotes the vector of partial derivatives $(\nabla_x \ell)_i = \partial \ell / \partial x_i$. If $y \in \mathcal{Y} = \mathbb{R}^m$ is another variable in the graph, which is computed directly from x according to $y = f(x)$, then the so-called *backpropagation map* for the function f is a linear map from $\nabla_y \ell \in \mathcal{Y}$ to $\nabla_x \ell \in \mathcal{X}$.

As explained in the Section 6.3, the backpropagation map is the linear map which is the adjoint of the differential. This property was also used by Ionescu et al. [99]. While they used the matrix inner product $\langle X, Y \rangle = \mathrm{tr}(X^T Y)$ to find the adjoint, we use Parseval's theorem, which states that the Fourier transform is unitary (except for a scale factor) and therefore preserves inner products $\langle x, y \rangle \propto \langle \widehat{x}, \widehat{y} \rangle$.

To find the linear map for backpropagation through the Correlation Filter, we first take the differentials of the system of equations in eq. 6.13 that defines the template w

$$\begin{cases} dk = \frac{1}{n}(dx \star x + x \star dx) \\ dk * \alpha + k * d\alpha = \frac{1}{n} dy \\ dw = d\alpha \star x + \alpha \star dx \end{cases} \tag{6.21}$$

and then take the Fourier transform of each equation and re-arrange to give the differential of each dependent variable in Figure 6.1b as a linear function (in the Fourier domain) of the differentials of its input variables

$$\begin{cases} \widehat{dk} = \frac{1}{n}(\widehat{dx}^* \circ \widehat{x} + \widehat{x}^* \circ \widehat{dx}) & \text{(6.22a)} \\ \widehat{d\alpha} = \widehat{k}^{-1} \circ \left[\frac{1}{n}\widehat{dy} - \widehat{dk} \circ \widehat{\alpha}\right] & \text{(6.22b)} \\ \widehat{dw} = \widehat{d\alpha}^* \circ \widehat{x} + \widehat{\alpha}^* \circ \widehat{dx} \ . & \text{(6.22c)} \end{cases}$$

Note that, while these are complex equations, that is simply because they are the Fourier transforms of real equations. The derivatives themselves are all computed with respect to real variables.

The adjoints of these linear maps define the overall backpropagation map from $\nabla_w \ell$ to $\nabla_x \ell$ and $\nabla_y \ell$. We defer the derivation to Appendix A.4 and present here the final result,

$$\begin{cases} \widehat{\nabla_\alpha \ell} = \widehat{x} \circ (\widehat{\nabla_w \ell})^* \\ \widehat{\nabla_y \ell} = \frac{1}{n}\widehat{k}^{-*} \circ \widehat{\nabla_\alpha \ell} \\ \widehat{\nabla_k \ell} = -\widehat{k}^{-*} \circ \widehat{\alpha}^* \circ \widehat{\nabla_\alpha \ell} \\ \widehat{\nabla_x \ell} = \widehat{\alpha} \circ \widehat{\nabla_w \ell} + \frac{2}{n}\widehat{x} \circ \mathrm{Re}\{\widehat{\nabla_k \ell}\} \ . \end{cases} \tag{6.23}$$

It is necessary to compute forward Fourier transforms at the start and inverse transforms at the end. The extension to multi-channel images is given in Appendix A.6.

As an interesting aside, we remark that, since we have the gradient of the loss with respect to the "desired" response y, it is possible to optimise for this parameter rather than specify it manually. However, in practice, we did not find learning this parameter to improve the tracking accuracy compared to the conventional choice of a fixed Gaussian response [25, 88].

6.5 EXPERIMENTS

The principal aim of our experiments is to investigate the effect of incorporating the Correlation Filter during the offline training of a CNN. We first compare against the symmetric Siamese architecture introduced in Chapter 5. We then compare the end-to-end trained CFNet to a variant where the features are replaced with features that were trained for a different task. Finally, we demonstrate that our method achieves strong results in real-time tracking.

6.5.1 *Implementation details*

We follow the same procedure adopted in Section 5 to minimise the loss (eq. 6.18) through SGD, with the Xavier parameters initialisation and using mini-batches of size 8. We use all the 3862 training videos of ImageNet Video [185], containing more than 1 million annotated frames, with multiple objects per frame. Training is conducted for 100 epochs, each sampling approximately 12 pairs (x'_i, z'_i) from each video, randomly extracted so that they are at most 100 frames apart.

During tracking, a Hann window is summed to the score map to penalise large displacements. Tracking in scale space is achieved by evaluating the network at the scale of the previous object and at one adjacent scale on either side, with a geometric step of 1.04. Updating the scale is discouraged by multiplying the responses of the scaled object by 0.97. To avoid abrupt transitions of object size, the scale is updated using a rolling average with a learning rate of 0.6. The aspect ratio of the reported bounding box, instead, is never modified. Unlike SiamFC, we update the target template (similarly to standard Correlation Filter trackers like KCF [88]) with a running average with a low learning rate (0.01). Although, the benefits of this update strategy are limited, as shown in Figure 6.3.

6.5.2 *Evaluation criteria*

Popular tracking benchmarks like VOT [113] and OTB [232, 233] have made all ground-truth annotations available and do not enforce train/validation/test splits. However, in order to avoid overfitting to the test set with design choices and hyperparameter selection, we consider OTB-2013, OTB-50 and OTB-100 as our *test set* and 129 videos from VOT-2014, VOT-2016 and TempleColor [129] as our *validation set*, excluding any videos which were already assigned to the test set. We perform all of our tracking experiments in Sections 6.5.3, 6.5.4 and 6.5.5 on the validation set with the same set of "natural" (tracking) hyperparameters described in Section 6.5.1, which are not tuned for any particular method.

As in the OTB benchmark [232, 233], we quantify the performance of the tracker on a sequence in terms of the average overlap (intersection over union, IOU) of the predicted and ground-truth rectangles in all frames. The success rate of a tracker at a given threshold τ corresponds to the fraction of frames in

which the overlap with the ground truth is at least τ. This rate is computed for a uniform range of 100 thresholds between 0 and 1, effectively constructing the cumulative distribution function. Trackers are compared using the area under this curve. Mimicking the TRE (Temporal Robustness Evaluation) mode of OTB, we choose three equispaced points per sequence and run the tracker from each until the end. Differently from the OTB evaluation, when the target is *lost* (i.e. the overlap with the ground truth becomes zero) the tracker is terminated and an overlap of zero is reported for all remaining frames.

Despite the large number of videos, we still find that, as the training of CFNet progresses, the tracking performance oscillates significantly. This is arguably due to the fact that the optimisation we conduct only captures the "matching" component of the overall tracking pipeline. To mitigate this effect, we average the final tracking results that are obtained using the parameters of the network at epochs 55, 60, ..., 95, 100 (the final epoch) to reduce the variance. These ten results are used to estimate the standard deviation of the distribution of results, providing error bars for most figures in this section. While it would be preferable to train all networks to convergence multiple times with different random seeds, this would require significantly more resources.

6.5.3 *Comparison to a fully-convolutional Siamese baseline*

Figures 6.2 and 6.3 report the accuracy of both CFNet and our Siamese baseline on the validation set for networks of varying depth. The feature extraction network of depth n is terminated after the n-th linear layer, including the following ReLU but not the following pooling layer (if any).

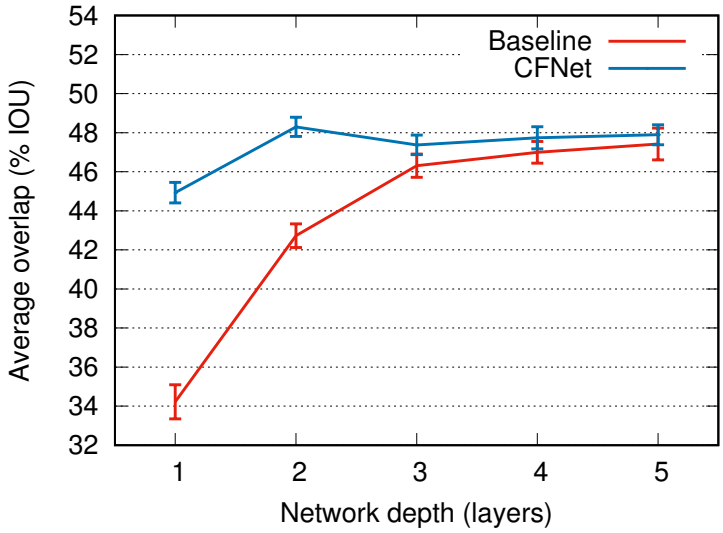
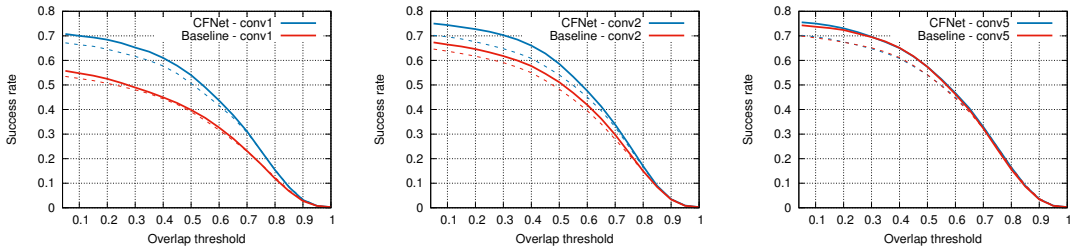


Figure 6.2: Tracker accuracy for different network depths, on the 129 videos of the validation set. Error bars indicate two standard deviations. Refer to section 6.5.3 for more details.



(a) **(b)** **(c)**

Figure 6.3: Success rates of rectangle overlap for individual trackers on the validation set. Solid and dotted lines represent methods that update the template with a running average learning rate of 0.01 and 0, respectively.

Our baseline diverges slightly from the network used in Chapter 5 in two ways. Firstly, we reduce the total stride of the network from 8 to 4 (2 at conv1, 2 at pool1) to avoid training Correlation Filters with small feature maps. Secondly, we always restrict the final layer to 32 output channels to preserve the high speed of the method with larger feature maps. These changes did not have a negative effect on the tracking performance of the baseline.

The results show that CFNet is significantly better than the baseline when shallow networks are used to compute features. Specifically, it brings a relative improvement of 31% and 13% for networks of depth one and two respectively.

However, at depths three, four and five, the difference is much less meaningful. CFNet is relatively unaffected by the depth of the network, whereas the performance of the baseline increases steadily and significantly with depth.

It seems that the ability of the Correlation Filter to adapt the representation to the content of the training image is less critical given a sufficiently expressive embedding function. In fact, the CF layer can be understood to encode prior knowledge of the test-time procedure. This prior may become redundant or even overly restrictive when enough model capacity and data are available.

Figure 6.3 additionally shows that updating the template is modestly helpful for both Baseline and CFnet architectures, at any depth. However, this is clearly more computationally expensive than the *no-update* strategy of Chapter 5, as CNN activations have to be computed for both x' and z' at every frame.

6.5.4 *Feature transfer experiment*

The motivation for this work was the hypothesis that incorporating the CF during training will result in features that are better suited to tracking with a CF. We now compare our end-to-end trained CFNet to variants that use features from alternative sources: *Baseline+CF* and *ImageNet+CF*. The results are presented in Figure 6.4.

To obtain the curve *Baseline+CF* we trained a baseline Siamese network of the desired depth and then combined those features with a CF during tracking. Results show that taking the CF into account during offline training is critical at depth one and two. However, it seems redundant when more convo-

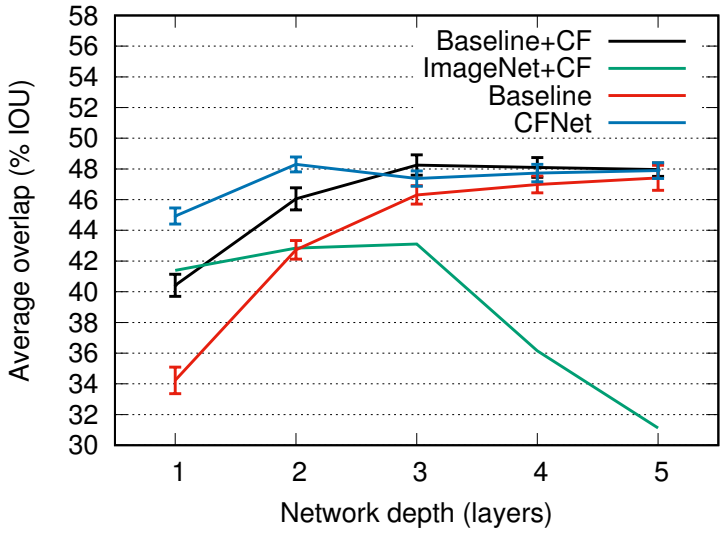


Figure 6.4: Accuracy of a Correlation Filter tracker when using features obtained via different methods. Error bars indicate two standard deviations. Refer to Section 6.5.4 for more details.

lutional layers are added, since using features from the *Baseline* in conjunction with the CF achieves similar performance.

The *ImageNet+CF* variant employs features taken from a network trained to solve the ImageNet classification challenge (as proposed by Sharif Razavian et al. [196]). The results show that these features, which are often the first choice for combining CFs with CNNs [44, 46, 136], are significantly worse than those learned by both CFNet and the fully-convolutional Siamese baseline. The particularly poor performance of these features at deeper layers is somewhat unsurprising since these layers are expected to have greater invariance to position when trained for classification.

6.5.5 *Importance of adaptation*

For a multi-channel CF, each channel p of the template w can be obtained as $w_p = \alpha \star x_p$, where α is itself a function of the exemplar x (Appendix A.5). To verify the importance of the online adaptation that solving a ridge regression

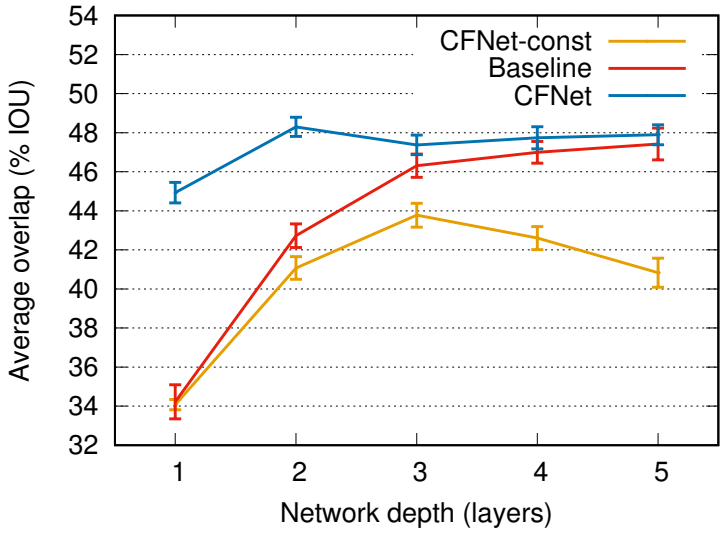


Figure 6.5: Comparison of CFNet to a "non-adaptive" variant of the architecture, in which the Lagrange multipliers do not depend on the image. Error bars indicate two standard deviations. Refer to Section 6.5.5 for more details.

problem at test-time should provide, we propose a "constant" version of the Correlation Filter (*CFNet-const*) where the vector of Lagrange multipliers α is instead a parameter of the network that is learned offline and remains fixed at test time.

Figure 6.5 compares CFNet to its constant variant. CFNet is consistently better, demonstrating that to improve over the baseline Siamese network it is paramount to backpropagate through the solution to the inverse convolution problem that defines the Lagrange multipliers.

6.5.6 *Speed and practical benefits*

The previous sections have demonstrated that there is a clear benefit to integrating Correlation Filters into Siamese networks when the feature extraction network is relatively shallow. Shallow networks are practically advantageous in that they require fewer operations and less memory to evaluate and store.

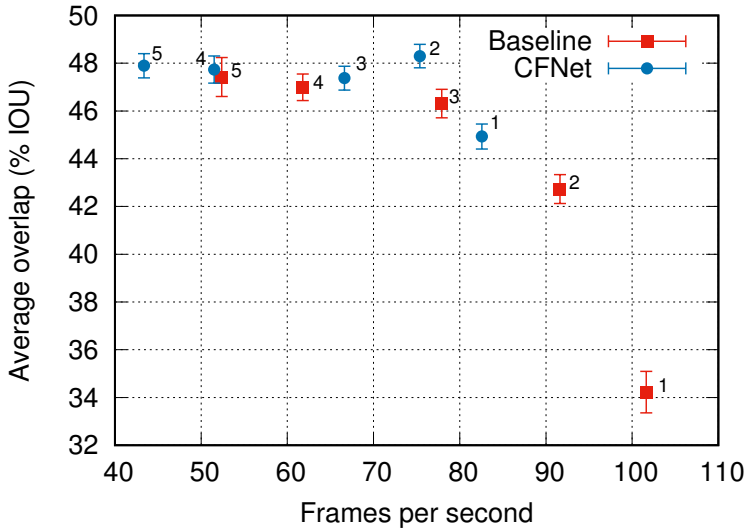


Figure 6.6: Tracker accuracy vs. speed for CFNet and Siamese baseline. Labels indicate network depth. CFNet enables better accuracy to be obtained at higher speeds using shallower networks. Error bars indicate two standard deviations. Refer to section 6.5.6 for more details.

In order to understand the trade-off, Figure 6.6 reports the speed and accuracy of both CFNet and the baseline for varying network depth[3].

This plot suggests that the two-layer CFNet could be the most interesting variant for practitioners requiring an accurate tracking algorithm that operates at high framerates. It runs at 75 frames per second and has less than 4% of the parameters of the five-layer baseline, requiring only 600kB to store. This characteristic may be of particular interest for embedded devices with limited memory. In contrast, methods like DeepSRDCF [44] and C-COT [46], which use out-of-the-box deep features for the Correlation Filter, run orders of magnitude slower. Even the one-layer CFNet remains competitive despite having less than 1% of the parameters of the five-layer baseline and requiring under 100kB to store.

3 The speed was measured using a 4.0GHz Intel i7 CPU and an NVIDIA Titan X GPU.

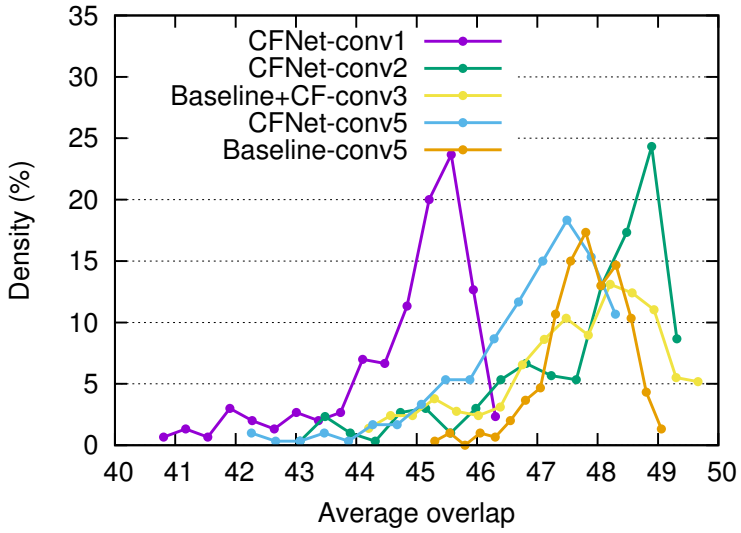


Figure 6.7: Empirical distribution of the average overlap for the hyperparameter search for CFNet and baselines. Refer to Section 6.5.7 for more details.

6.5.7 *Hyperparameter optimisation*

The hyperparameters that define the simplistic tracking algorithm have a significant impact on tracking accuracy. Choosing hyperparameters is a difficult optimisation problem: we cannot use gradient descent because the function is highly discontinuous, and each function evaluation is expensive because it involves running a tracker on every sequence from multiple starting points.

For the experiments described so far, where we sought to make a fair comparison of different architectures, we therefore used a *natural* choice of hyperparameters that were not optimised for any particular architecture. Ideally, we would use the optimal hyperparameters for each variant, but it would have been computationally prohibitive.

To compare our methods against the state of the art, however, we optimise the parameters (of the tracking algorithm only) on a held-out validation set. We use random search with a uniform distribution on a reasonable range for each hyperparameter. Specifically, we sample 300 random vectors of hyperparameters and run the OTB-TRE evaluation (with 3 subsequences per video)

on the 129 videos of our validation set. Each method is then evaluated once on the test sets (OTB-2013, OTB-50 and OTB-100) using the hyperparameter vector which gave the best results on the validation set. Even though the ground-truth labels are available for the videos in the benchmarks, we do not choose hyperparameters to optimise the results on the benchmarks, as this would not give a meaningful estimate of the generalisation ability of the method.

Note that this random search is performed after training and is only used to choose parameters for the online tracking algorithm. The same network is used for all random samples, chose at the training epoch with the best tracking results on the validation set (with natural tracking parameters).

Figure 6.7 shows, for each method, the empirical distribution of results (in terms of average overlap) that is induced by the distribution of tracking parameters during the random search. Interestingly, while the best average overlap for CFNet-conv1 is less than that of Baseline-conv5 (38.3% vs. 39.1%), the mean is actually higher (36.4% vs. 35.3%) and the distribution seems to assign more mass to better results.

6.5.8 *Comparison with the state of the art*

We use the OTB-2013/50/100 benchmarks to confirm that our results are on par with the state of the art. All numbers in this section are obtained using the OTB toolkit [232]. We report the results for the three best instantiations of CFNet from Figure 6.4 (*CFNet-conv2*, *CFNet-conv5*, *Baseline+CF-conv3*), the best variant of the baseline (*Baseline-conv5*) and the single-layer network (*CFNet-conv1*). We compare our methods against state-of-the-art trackers that

		OTB-2013				OTB-50				OTB-100			
		OPE		TRE		OPE		TRE		OPE		TRE	
Method	**speed** (fps.)	IOU	prec.	IOU	prec.	IOU	prec.	IOU	prec.	IOU	prec.	IOU	prec.
CFNet-conv1	83	57.8	71.4	58.6	71.7	48.8	61.3	51.0	63.6	53.6	65.8	55.9	67.6
CFNet-conv2	75	61.1	74.6	**64.0**	**77.9**	53.0	66.0	56.5	70.2	56.8	69.3	60.6	73.2
Baseline+CF-conv3	67	61.0	74.8	63.1	76.8	53.8	66.5	**57.4**	**70.8**	**58.9**	71.1	61.1	73.4
CFNet-conv5	43	61.1	73.6	62.6	75.7	**53.9**	**67.0**	56.6	70.1	58.6	71.1	60.8	72.7
Baseline-conv5	52	**61.8**	75.3	**64.0**	77.3	51.7	64.1	56.1	69.1	58.8	71.4	**61.6**	**73.7**
SiamFC-3s [19]		60.7	73.5	61.8	75.0	51.6	63.9	55.5	69.2	58.2	70.2	60.5	72.8
Staple [18]		60.0	72.5	61.7	74.2	50.9	63.4	54.1	67.5	58.1	**71.6**	60.4	72.8
LCT [137]		61.2	**78.0**	59.4	74.2	49.2	62.5	49.5	61.7	56.2	69.2	56.9	68.2
SAMF [128]		–	–	–	–	46.2	60.7	51.4	65.6	53.9	69.0	57.7	71.4
DSST [42]		55.4	67.5	56.6	68.4	45.2	56.6	48.4	60.1	51.3	63.1	–	–

Table 6.1: Perfomance as overlap (IOU) and precision produced by the OTB toolkit for the OTB-2013, OTB-50 and OTB-100 datasets. The **first** and second best results are highlighted in each column. For details refer to Section 6.5.8.

can operate in real-time: SiamFC-3s [19], Staple [18] and LCT [137]. We also include the recent SAMF [128] and DSST [42] for reference.

Both overlap (IOU) and precision scores [233] are reported for OPE and TRE (temporal robustness) evaluations described in Section 4.4.

Similarly to the analysis on the validation set, *CFNet-conv2* is among the top performers and its accuracy rivals that of *Baseline-conv5*, which possesses approximately 30× as many parameters. In general, our best proposed CFNet variants are superior (albeit modestly) to the state of the art. In order to focus on the impact of our contribution, we decided to avoid including orthogonal improvements which can often be found in the tracking literature (e.g. bounding-box regression [127], ensembling of multiple cues [18, 137], data augmentation [247], optical flow [206], *etc.*).

6.6 DISCUSSION

In this chapter, we proposed the Correlation Filter network, an asymmetric architecture that backpropagates gradients through an online learning algorithm to optimise the underlying feature representation. This is made feasible by establishing an efficient backpropagation map for the solution to a system of circulant equations and by training the overall system in a meta-learning framework (Section 3.1).

Our empirical investigation provides several insights. *a)* Unsurprisingly, end-to-end training is important to learn representations tailored for the Correlation Filter; *b)* unlike the symmetric fully-convolutional network of Chapter 5, CFNet accuracy does not improve when the network's capacity increases; *c)* our method has the practical advantage of allowing lightweight networks to achieve strong results.

7

CONCLUSION

Machine learning algorithms (and deep neural networks in particular) often thrive when trained with large amounts of data. However, it is not uncommon to encounter situations in which data is scarce. Motivated by the importance of these scenarios, in this thesis we proposed several novel methods specifically designed to operate under restrictive data regimes. In particular, we considered the two computer vision problems of arbitrary object tracking and few-shot classification, as they have an important practical relevance.

In Chapter 4, we introduced *Staple*, a tracking method that, by exploiting the complementarity of two simple and efficient linear models, achieves good robustness to challenging tracking sequences and runs at a high speed. *Staple* is competitive with much more sophisticated trackers that make use of DNN embeddings [44, 94, 227], while being considerably faster. Therefore, at least for the problem of arbitrary object tracking, it is still valuable to experiment with traditional models and features. However, a glaring limitation of *Staple* and many other established tracking methods is that they are constrained by their design to only use the training data available online. This means that the wealth of information represented by vast video datasets such as YouTube Bounding-Boxes [176] or ImageNet VID [185] cannot be leveraged.

To address this problem, in Chapter 5 we introduced SiamFC, a tracking framework based on a fully-convolutional Siamese network that significantly differs from the widely adopted *online learning* approach (Section 2.1), as it considers tracking as the iterative evaluation of an offline-trained deep simi-

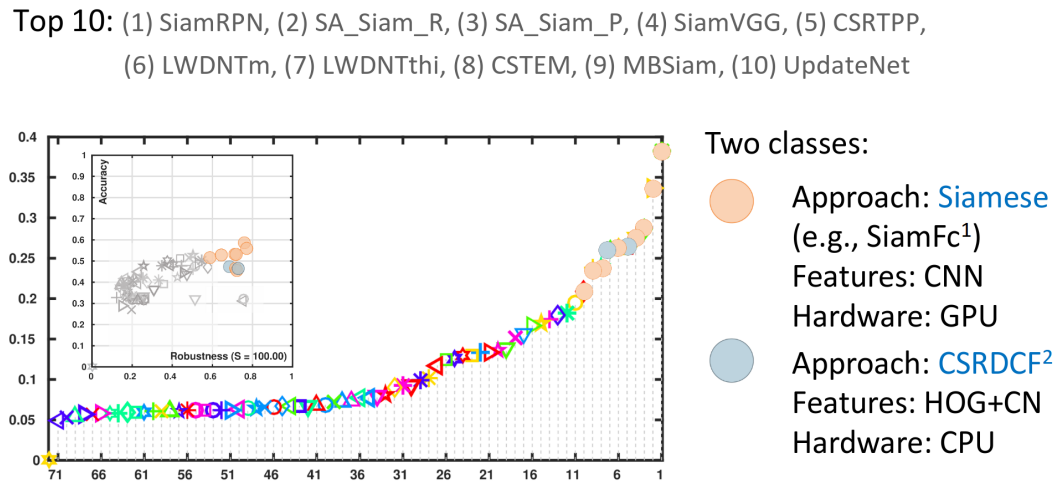


Figure 7.1: EAO (y-axis) and ranking (x-axis) of the VOT-2018 real-time challenge. Eight out of the ten best-performing methods stem from the fully-convolutional Siamese approach discussed in Chapter 5. Figure from the presentation of the VOT-2018 committee [115].

larity function. The original paper from which Chapter 5 stems [19] has generated a significant interest in the tracking community. At the recent 2018 edition of the VOT challenge [115], eight out of the top ten performing methods in the real-time track were based on our approach, as shown in Figure 7.1. Moreover, although SiamFC was originally presented more than two years ago [19], it still ranks among the very top methods in the majority of the new tracking benchmarks that have been presented in 2018: TLP [148], OxUvA [219], TrackingNet [151], VOT-LT [134], LaSoT [58] and GOT-10k [97].

Despite their efficacy and efficiency, SiamFC and deep similarity functions have, in a sense, the *opposite* problem to traditional tracking methods: they only allow parametric adaptation *offline*. Instead, it can be beneficial to modify a model online to make good use of the new (albeit limited) data provided. For this reason, in this thesis we also presented two novel approaches that can learn how to learn from limited data. By organising the training process in two levels (Section 3.1), it is possible to quickly train learners to specialise to different small learning problems (e.g. Chapter 3, Chapter 6 and [64, 174, 222]). In such a way, an iterative optimiser like SGD can be used to learn from a

multitude of small problems and thus progressively shift the inductive bias of the base learners [220]. This can happen, for example, by learning the features or the hyperparameters used by the base learner.

In this context, it became apparent that, while problem-specific parametric adaptation is a desirable capability, it is not obviously beneficial in terms of performance over more straightforward and non-adaptive similarity learning approaches. For example, with deep enough neural networks, CFNet (Chapter 6) does not outperform its baseline SiamFC (Chapter 5). Moreover, in few-shot classification benchmarks, similarity-based methods can achieve high performance [72, 199, 202]. Overall, when training data specific to a particular object is not available, similarity learning turns out to be a simple yet effective strategy. Nonetheless, it is important to limit the validity of this observation with respect to the specific computer vision problems and datasets taken into account.

Throughout this thesis, considerable attention has been paid to computational efficiency, often achieved by exploiting the "shape" of the data at hand. For example, both Correlation Filters (Chapters 4 and 6) and fully-convolutional Siamese networks (Chapters 5 and 6) owe their speed to the fact that, in the tracking scenario, candidates are represented by overlapping windows in a video frame. Moreover, thanks to the Woodbury identity [167], the computational cost of the algorithms for few-shot classification presented in Chapter 3 only grows linearly (rather than quadratically) with the embedding size.

We believe that several interesting research directions can arise from this thesis. Some of them (based on our preliminary results) have already led to papers that have been published recently (e.g. [52, 84, 127, 238, 245, 247]).

Online update during tracking. For instance, one aspect that has been overlooked throughout this thesis is how to update the target representation during tracking. Interestingly, the offline-learnt similarity function we introduced is robust to appearance changes without any online adaptation. In Chapter 5, we speculated that this is probably due to the training strategy used, which exposes the network to frames that are on average a couple of seconds apart. We also observed how this approach has the undesirable side effect of making the tracker less robust in scenes in which some objects look similar to the target.

In Chapter 6, we showed that the simplistic update strategy of maintaining a running average of target templates is only marginally preferable to keeping the same appearance throughout the entire video sequence. Hence, a valuable research avenue could be to investigate ways to better leverage the large amount of video data currently available [176, 185, 236], such that the object representation's update could be seamlessly integrated into the offline training process. In particular, since Siamese architectures can be interpreted as unrolled recurrent neural networks trained on short sequences of two frames, an interesting endeavour could be to extend the training scope beyond this limit, perhaps using the embedding obtained in Chapter 5 as a strong initialisation.

Losses for the fully-convolutional Siamese framework. Another aspect that could be further investigated is the type of loss used to train fully-

convolutional Siamese networks. Whilst the vanilla logistic loss we adopted achieves good results, the metric-learning literature is full of promising alternatives (e.g. [38, 74, 91, 149]) that are widely used in analogous computer vision tasks, such as face recognition and person re-identification. Moreover, in Chapters 5 and 6, we limited ourselves to the background surrounding the target object as the source for negative samples. Instead, during training, it could be important to expose the network to increasingly "harder" negative samples (i.e. *hard negative mining*), which might curb the predisposition of the network to erroneously "fire" on objects that are similar to the target.

Trends in arbitrary object tracking. Naturally, tracking algorithms have been constrained by the specific scenarios portrayed by the most popular tracking benchmarks [113, 198, 233]. While offering a wide range of challenging situations, the number of test sequences in these benchmarks is limited to a few tens. Moreover, sequences are relatively short, with the target always being at least partially visible. Finally, these benchmarks have not established an explicit separation between training, validation and test sets; while training on the test set is clearly unacceptable, there has not been much consensus around what set should be used for hyperparameter optimisation. Arguably, excessively consulting the test set to perform design decisions and choose hyperparameters is a form of overfitting that results in higher performance on the targeted set but can lower the generalisation capability.

A recent wave of new tracking benchmarks [58, 97, 134, 148, 151, 219] aims at addressing the issues mentioned above. They offer much larger datasets, longer sequences, labelled disappearances of the target object and dataset splits. By leveraging large and diverse datasets, and adopting benchmarks with stricter evaluation protocols, we believe that the tracking community will make significant leaps forward.

What is more, several recent works on semi-supervised object segmentation [165] have presented fast methods (e.g. [160, 228, 237]) that operate in a scenario similar to the one of arbitrary object tracking, suggesting that in the future these two problems could be considered jointly [228].

Data domains. Different researchers collect datasets for different purposes, which thus, inevitably, exhibit very distinctive biases [213]. For this reason, different datasets have often been considered as being entirely different *domains*. In particular, the machine learning subcommunity working on *domain adaptation* is focused on designing methods that can apply the knowledge obtained from a *source* domain to a *target* one [40]. Similar to the case of few-shot classification, the motivation behind this learning paradigm is that we can choose a source domain with plenty of labelled data and then *transfer* what we have learnt to another domain that might be plagued by data scarcity. Nonetheless, since we preferred to focus on proposing novel techniques, in this thesis we adopted the consolidated problem setups of few-shot classification and object tracking and left considerations on domain differences and datasets bias to future work.

In few-shot classification, for instance, despite the sets of classes between training and test splits being disjoint, they still belong to the same dataset. It would be valuable to understand if, in a more challenging scenario in which both classes *and* domains are disjoint, methods that perform adaptation at test time have significant edge over simpler similarity-based approaches. Two collections of datasets that could allow this analysis are the Visual Decathlon of Rebuffi et al. [177] and the Meta-dataset of Triantafillou et al. [214].

Similar arguments can be made for the task of arbitrary object tracking. While training the models presented in Chapters 5 and 6, we have been constrained to use the only large labelled video dataset available at the time of writing: ImageNet VID [185]. However, this dataset is profoundly dissimilar to

tracking benchmarks such as VOT [113] or OTB [233]. For instance, it does not contain targets of the class "person", but only animals and vehicles; moreover, objects often exhibit limited movements, while in tracking benchmarks they usually perform fast movements and undergo abrupt appearance changes.

Given the recent abundance of video data [176, 236], a simple but promising strategy to improve tracking performance could be to characterise the domain represented by the sequences belonging to the OTB and VOT benchmarks and then use training data that is more closely related to such a domain. However, since situations of domain discrepancy between training and test sets are often unavoidable, an interesting research path could be to design online techniques (e.g. [156, 163]) that are capable of adapting to the type of data encountered at test time.

A

APPENDIX

A.1 DIFFERENT GAUSSIAN PRIORS FOR REGULARISATION

In a ridge regression problem (one of the base learners proposed in Chapter 3), the regularisation term can be seen as a prior Gaussian distribution of the parameters in a Bayesian interpretation. In the most common case of λI, it corresponds to an isotropic Gaussian prior on the parameters [207].

In addition to the case in which λ is a scalar, we experiment with the variant $\text{diag}(\boldsymbol{\lambda})$, corresponding to an axis-aligned Gaussian prior with an independent variance for each parameter, which can potentially exploit the fact that the parameters have different scales. Replacing λI with $\text{diag}(\boldsymbol{\lambda})$ in eq. 3.4, the final expression for W after having applied the Woodbury identity becomes:

$$W = \Lambda(Z) = \text{diag}(\boldsymbol{\lambda})^{-1} X^T (X \text{diag}(\boldsymbol{\lambda})^{-1} X^T + I_{n,n})^{-1} Y. \tag{A.1}$$

A.2 BASE LEARNER HYPERPARAMETERS

Figure A.1 illustrates the effect of using SGD to learn, together with the parameters ω of the CNN, also the hyperparameters (ρ in eq. 3.2, Section 3.1) of the base learner Λ. We find that it is very important to learn the scalar α (right) used to calibrate the output of R2-D2 in eq. 3.6 (Section 3.2), while it is indifferent whether or not to learn λ. Note that, by using SGD to update α,

it is possible (e.g. in the range $[10^{-3}, 10^{0}]$) to recover from poor initial values and suffer just a little performance loss w.r.t. the optimal value of $\alpha = 10$.

The left plot of Figure A.1 also shows the performance of R2-D2 with the variant diag(λ) introduced in Appendix A.1. Unfortunately, despite this formulation allows us to make use of a more expressive prior, it does not improve the results compared to using a simple scalar λ. Moreover, performance abruptly deteriorate for $\lambda > 0.01$.

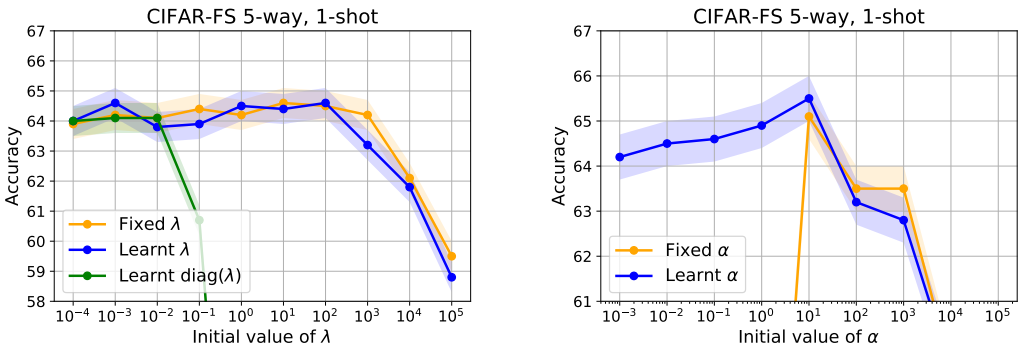


Figure A.1: Sensitivity of R2-D2 to (fixed and meta-learnt) hyperparameters. Shaded areas represent 95% confidence intervals.

A.3 CIFAR-FS SPLITS

To facilitate experiments reproducibility, below we report the dataset splits used for the experiment on CIFAR in Chapter 3.

Meta-training classes: train, skyscraper, turtle, raccoon, spider, orange, castle, keyboard, clock, pear, girl, seal, elephant, apple, aquarium fish, bus, mushroom, possum, squirrel, chair, tank, plate, wolf, road, mouse, boy, shrew, couch, sunflower, tiger, caterpillar, lion, streetcar, lawn mower, tulip, forest, dolphin, cockroach, bear, porcupine, bee, hamster, lobster, bowl, can, bottle, trout, snake, bridge, pine tree, skunk, lizard, cup, kangaroo, oak tree, dinosaur, rabbit, orchid, willow tree, ray, palm tree, mountain, house, cloud.

Meta-validation classes: otter, motorcycle, television, lamp, crocodile, shark, butterfly, beaver, beetle, tractor, flatfish, maple tree, camel, crab, sea, cattle.

Meta-test classes: baby, bed, bicycle, chimpanzee, fox, leopard, man, pickup truck, plain, poppy, rocket, rose, snail, sweet pepper, table, telephone, wardrobe, whale, woman, worm.

A.4 BACKPROPAGATION FOR THE CORRELATION FILTER

As described in Section 6.3, the backpropagation map is the adjoint of the linear maps that is the differential. These linear maps for the Correlation Filter are presented in eq. 6.22. We are free to obtain these adjoint maps in the Fourier domain since Parseval's theorem provides the preservation of inner products. Let J_1 denote the map $dx \mapsto dk$ in eq. 6.22a. Hence, manipulation of the inner product

$$\begin{aligned}\langle Fdk, FJ_1(dx)\rangle &= \left\langle \widehat{dk}, \tfrac{1}{n}(\widehat{dx}^* \circ \widehat{x} + \widehat{x}^* \circ \widehat{dx})\right\rangle \\ &= \tfrac{1}{n}\left[\langle \widehat{dx}, \widehat{dk}^* \circ \widehat{x}\rangle + \langle \widehat{dk} \circ \widehat{x}, \widehat{dx}\rangle\right] \\ &= \left\langle \widehat{dx}, \tfrac{2}{n}\operatorname{Re}\{\widehat{dk}\} \circ \widehat{x}\right\rangle \end{aligned} \tag{A.2}$$

gives the backpropagation map

$$\widehat{\nabla_x \ell} = \tfrac{2}{n}\widehat{x} \circ \operatorname{Re}\{\widehat{\nabla_k \ell}\} \ . \tag{A.3}$$

Similarly, for the linear map $dk, dy \mapsto d\alpha$ in eq. 6.22b,

$$\begin{aligned}\langle Fd\alpha, FJ_2(dk, dy)\rangle &= \left\langle \widehat{d\alpha}, \widehat{k}^{-1}[\tfrac{1}{n}\widehat{dy} - \widehat{dk} \circ \widehat{\alpha}]\right\rangle \\ &= \left\langle \tfrac{1}{n}\widehat{k}^{-*} \circ \widehat{d\alpha}, \widehat{dy}\right\rangle + \left\langle -\widehat{k}^{-*} \circ \widehat{\alpha}^* \circ \widehat{d\alpha}, \widehat{dk}\right\rangle \ , \end{aligned} \tag{A.4}$$

the backpropagation maps are

$$\widehat{\nabla_y \ell} = \tfrac{1}{n} \widehat{k}^{-*} \circ \widehat{\nabla_\alpha \ell} \tag{A.5}$$

$$\widehat{\nabla_k \ell} = -\widehat{k}^{-*} \circ \widehat{\alpha}^* \circ \widehat{\nabla_\alpha \ell} \ , \tag{A.6}$$

and for the linear map $dx, d\alpha \mapsto dw$ in eq. 6.22c,

$$\begin{aligned} \langle F dw, F J_3(dx, d\alpha) \rangle &= \langle \widehat{dw}, \widehat{d\alpha}^* \circ \widehat{x} + \widehat{\alpha}^* \circ \widehat{dx} \rangle \\ &= \langle \widehat{d\alpha}, \widehat{dw}^* \circ \widehat{x} \rangle + \langle \widehat{dw} \circ \widehat{\alpha}, \widehat{dx} \rangle \ , \end{aligned} \tag{A.7}$$

the backpropagation maps are

$$\widehat{\nabla_\alpha \ell} = \widehat{x} \circ (\widehat{\nabla_w \ell})^* \ , \tag{A.8}$$

$$\widehat{\nabla_x \ell} = \widehat{\alpha} \circ \widehat{\nabla_w \ell} \ . \tag{A.9}$$

The two expressions for $\nabla_x \ell$ above are combined to give the backpropagation map for the entire Correlation Filter block in eq. 6.23.

A.5 MULTI-CHANNEL CORRELATION FILTER

There is little advantage to the dual solution when training a single-channel Correlation Filter from the circular shifts of a single base example. However, the dual formulation is much more efficient in the multi-channel case [88].

For signals with k channels, each multi-channel signal is a collection of scalar-valued signals $x = (x_1, \ldots, x_k)$, and the data term becomes

$$\| \sum_p w_p \star x_p - y \|^2 = \| \sum_p X_p^T w_p - y \|^2 \tag{A.10}$$

and each channel of the template is obtained from the dual variables

$$w_p = X_p \alpha = \alpha \star x_p \tag{A.11}$$

The solution to the dual problem is still $\alpha = \frac{1}{n} K^{-1} y$, however the kernel matrix is now given

$$K = \frac{1}{n} \sum_p X_p^T X_p + \lambda I \tag{A.12}$$

and the linear map defined by this matrix is equivalent to convolution with the signal

$$k = \frac{1}{n} \sum_p x_p \star x_p + \lambda \delta \ . \tag{A.13}$$

Therefore the solution is defined by the equations

$$\begin{cases} k = \frac{1}{n} \sum_p x_p \star x_p + \lambda \delta \\ k * \alpha = \frac{1}{n} y \\ w_p = \alpha \star x_p \quad \forall p \end{cases} \tag{A.14}$$

and the template can be computed efficiently in the Fourier domain

$$\begin{cases} \widehat{k} = \frac{1}{n} \sum_p \widehat{x}_p^* \circ \widehat{x}_p + \lambda \mathbb{1} \\ \widehat{\alpha} = \frac{1}{n} \widehat{k}^{-1} \circ \widehat{y} \\ \widehat{w}_p = \widehat{\alpha}^* \circ \widehat{x}_p \quad \forall p \ . \end{cases} \tag{A.15}$$

It is critical that the computation scales only linearly with the number of channels.

A.6 BACKPROPAGATION FOR THE MULTI-CHANNEL CORRELATION FILTER

The differentials of the equations that define the multi-channel CF in eq. A.14 are

$$\begin{cases} dk = \frac{1}{n}\sum_p (dx_p \star x_p + x_p \star dx_p) \\ dk * \alpha + k * d\alpha = \frac{1}{n} dy \\ dw_p = d\alpha \star x_p + \alpha \star dx_p \quad \forall p \ , \end{cases} \tag{A.16}$$

and taking the Fourier transforms of these equations gives

$$\begin{cases} \widehat{dk} = \frac{1}{n}\sum_p \left(\widehat{dx}_p^* \circ \widehat{x}_p + \widehat{x}_p^* \circ \widehat{dx}_p\right) \\ \widehat{d\alpha} = \widehat{k}^{-1} \circ \left[\frac{1}{n}\widehat{dy} - \widehat{dk} \circ \widehat{\alpha}\right] \\ \widehat{dw}_p = \widehat{d\alpha}^* \circ \widehat{x}_p + \widehat{\alpha}^* \circ \widehat{dx}_p \quad \forall p \ . \end{cases} \tag{A.17}$$

Now, to find the adjoint of the map $dx \mapsto dk$, we re-arrange the inner product

$$\begin{aligned} \langle Fdk, FJ_1(dx)\rangle &= \left\langle \widehat{dk}, \tfrac{1}{n}\sum_p \left(\widehat{dx}_p^* \circ \widehat{x}_p + \widehat{x}_p^* \circ \widehat{dx}_p\right)\right\rangle \\ &= \tfrac{1}{n}\sum_p \left[\langle \widehat{dx}_p, \widehat{dk}^* \circ \widehat{x}_p\rangle + \langle \widehat{dk} \circ \widehat{x}_p, \widehat{dx}_p\rangle\right] \\ &= \sum_p \langle \widehat{dx}_p, \tfrac{2}{n}\,\mathrm{Re}\{\widehat{dk}\} \circ \widehat{x}_p\rangle \end{aligned} \tag{A.18}$$

to give the backpropagation map

$$\widehat{\nabla_{x_p}\ell} = \tfrac{2}{n}\widehat{x}_p \circ \mathrm{Re}\{\widehat{\nabla_k \ell}\} \quad \forall p \ . \tag{A.19}$$

The linear map $dk, dy \mapsto d\alpha$ is identical to the single-channel case. To find the adjoint of the map $dx, d\alpha \mapsto dw$, we examine the inner-product

$$\begin{aligned}\langle dw, J_3(dx, d\alpha)\rangle &= \textstyle\sum_p \langle \widehat{dw}_p, \widehat{d\alpha}^* \circ \widehat{x}_p + \widehat{\alpha}^* \circ \widehat{dx}_p\rangle \\ &= \left\langle \widehat{d\alpha}, \textstyle\sum_p \widehat{dw}_p^* \circ \widehat{x}_p \right\rangle + \textstyle\sum_p \langle \widehat{dw}_p \circ \widehat{\alpha}, \widehat{dx}_p\rangle \ , \end{aligned} \tag{A.20}$$

giving the backpropagation maps

$$\widehat{\nabla_\alpha \ell} = \textstyle\sum_p \widehat{x}_p \circ (\widehat{\nabla_{w_p} \ell})^* \ , \tag{A.21}$$

$$\widehat{\nabla_{x_p} \ell} = \widehat{\alpha} \circ \widehat{\nabla_{w_p} \ell} \quad \forall p \ . \tag{A.22}$$

Finally, combining these results gives the procedure for backpropagation in the multi-channel case

$$\begin{cases} \widehat{\nabla_\alpha \ell} = \sum_p \widehat{x}_p \circ (\widehat{\nabla_{w_p} \ell})^* \\ \widehat{\nabla_y \ell} = \frac{1}{n} \widehat{k}^{-*} \circ \widehat{\nabla_\alpha \ell} \\ \widehat{\nabla_k \ell} = -\widehat{k}^{-*} \circ \widehat{\alpha}^* \circ \widehat{\nabla_\alpha \ell} \\ \widehat{\nabla_{x_p} \ell} = \widehat{\alpha} \circ \widehat{\nabla_{w_p} \ell} + \frac{2}{n} \widehat{x}_p \circ \mathrm{Re}\{\widehat{\nabla_k \ell}\} \quad \forall p \ . \end{cases} \tag{A.23}$$

BIBLIOGRAPHY

[1] Mozilla commonvoice project. https://voice.mozilla.org/en. (Cited on page 3.)

[2] S. Abu-El-Haija, N. Kothari, J. Lee, P. Natsev, G. Toderici, B. Varadarajan, and S. Vijayanarasimhan. Youtube-8m: A large-scale video classification benchmark. *arXiv preprint arXiv:1609.08675*, 2016. (Cited on page 3.)

[3] A. Adam, E. Rivlin, and I. Shimshoni. Robust fragments-based tracking using the integral histogram. In *IEEE Conference on Computer Vision and Pattern Recognition*, 2006. (Cited on page 15.)

[4] H. Altae-Tran, B. Ramsundar, A. S. Pappu, and V. Pande. Low data drug discovery with one-shot learning. *ACS central science*, 2017. (Cited on page 3.)

[5] M. Andrychowicz, M. Denil, S. Gomez, M. W. Hoffman, D. Pfau, T. Schaul, and N. de Freitas. Learning to learn by gradient descent by gradient descent. In *Advances in Neural Information Processing Systems*, 2016. (Cited on pages 22, 27, and 28.)

[6] A. Argyriou, T. Evgeniou, and M. Pontil. Multi-task feature learning. In *Advances in Neural Information Processing Systems*, 2007. (Cited on page 27.)

[7] S. Avidan. Support vector tracking. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2004. (Cited on page 16.)

[8] S. Avidan. Ensemble Tracking. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2007. (Cited on page 16.)

[9] B. Babenko, M.-H. Yang, and S. Belongie. Visual tracking with online multiple instance learning. In *IEEE Conference on Computer Vision and Pattern Recognition*, 2009. (Cited on pages 8, 9, and 16.)

[10] B. Babenko, M.-H. Yang, and S. Belongie. Robust object tracking with online multiple instance learning. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2011. (Cited on page 47.)

[11] S. Baker and I. Matthews. Lucas-kanade 20 years on: A unifying framework. *International Journal of Computer Vision*, 2004. (Cited on page 15.)

[12] P. Baldi, P. Sadowski, and D. Whiteson. Searching for exotic particles in high-energy physics with deep learning. *Nature communications*, 5:4308, 2014. (Cited on page 1.)

[13] N. M. Ball and R. J. Brunner. Data mining and machine learning in astronomy. *International Journal of Modern Physics D*, 2010. (Cited on page 1.)

[14] E. Bart and S. Ullman. Cross-generalization: Learning novel classes from a single example by feature replacement. In *IEEE Conference on Computer Vision and Pattern Recognition*, 2005. (Cited on page 20.)

[15] S. Bengio, Y. Bengio, J. Cloutier, and J. Gecsei. On the optimization of a synaptic learning rule. In *Preprints Conf. Optimality in Artificial and Biological Neural Networks*, pages 6–8. Univ. of Texas, 1992. (Cited on pages 21 and 22.)

[16] L. Bertinetto, J. Valmadre, S. Golodetz, O. Miksik, and P. H. S. Torr. The importance of estimating object extent when tracking with correlation filters. *Technical report*, 2015.

[17] L. Bertinetto*, J. F. Henriques*, J. Valmadre*, P. H. S. Torr, and A. Vedaldi. Learning feed-forward one-shot learners. In *Advances in Neural Information Processing Systems*, 2016. (Cited on pages 4, 22, 26, and 91.)

[18] L. Bertinetto, J. Valmadre, S. Golodetz, O. Miksik, and P. H. S. Torr. Staple: Complementary learners for real-time tracking. In *IEEE Conference on Computer Vision and Pattern Recognition*, 2016. (Cited on pages 10, 15, 57, 77, 78, 89, 112, and 113.)

[19] L. Bertinetto*, J. Valmadre*, J. F. Henriques, A. Vedaldi, and P. H. S. Torr. Fully-convolutional siamese networks for object tracking. In *European Conference on Computer Vision workshops*, 2016. (Cited on pages 10, 19, 81, 83, 88, 90, 99, 112, 113, and 115.)

[20] L. Bertinetto, J. F. Henriques, P. H. S. Torr, and A. Vedaldi. Meta-learning with differentiable closed-form solvers. In *International Conference on Learning Representations*, 2019. (Cited on pages 9 and 91.)

[21] C. Bibby and I. Reid. Robust Real-Time Visual Tracking using Pixel-Wise Posteriors. In *European Conference on Computer Vision*, 2008. (Cited on pages 15 and 55.)

[22] C. M. Bishop. *Pattern Recognition and Machine Learning (Information Science and Statistics)*. Springer-Verlag, Berlin, Heidelberg, 2006. (Cited on pages 1, 2, and 38.)

[23] M. J. Black and A. D. Jepson. Eigentracking: Robust matching and tracking of articulated objects using a view-based representation. *International Journal of Computer Vision*, 1998. (Cited on page 16.)

[24] V. N. Boddeti, T. Kanade, and B. V. K. Kumar. Correlation Filters for Object Alignment. In *IEEE Conference on Computer Vision and Pattern Recognition*, 2013. (Cited on page 45.)

[25] D. S. Bolme, J. R. Beveridge, B. A. Draper, and Y. M. Lui. Visual object tracking using adaptive correlation filters. In *IEEE Conference on Computer Vision and Pattern Recognition*, 2010. (Cited on pages 8, 17, 45, 50, 89, 90, 99, and 102.)

[26] L. Bottou. Large-scale machine learning with stochastic gradient descent. In *Proceedings of COMPSTAT'2010*. Springer, 2010. (Cited on page 2.)

[27] K. Briechle and U. D. Hanebeck. Template matching using fast normalized cross correlation. In *Optical Pattern Recognition XII*. International Society for Optics and Photonics, 2001. (Cited on page 15.)

[28] J. Bromley, J. W. Bentz, L. Bottou, I. Guyon, Y. LeCun, C. Moore, E. Säckinger, and R. Shah. Signature verification using a "Siamese" time delay neural network. *International Journal of Pattern Recognition and Artificial Intelligence*, 1993. (Cited on pages 20, 26, and 71.)

[29] K. P. Burnham and D. R. Anderson. *Model selection and multimodel inference: a practical information-theoretic approach*. Springer Science & Business Media, 2003. (Cited on page 4.)

[30] Z. Cai, L. Wen, Z. Lei, N. Vasconcelos, and S. Z. Li. Robust deformable and occluded object tracking with dynamic graph. *IEEE Transactions on Image Processing*, 2014. (Cited on pages 47 and 58.)

[31] S. Carey. Beyond fast mapping. *Language Learning and Development*, 2010. (Cited on pages 5 and 19.)

[32] R. Caruana. Multitask learning. In *Learning to learn*. Springer, 1998. (Cited on page 27.)

[33] S. Challa, M. R. Morelande, D. Mušicki, and R. J. Evans. *Fundamentals of object tracking*. Cambridge University Press, 2011. (Cited on pages 3 and 7.)

[34] K. Chen and W. Tao. Once for all: a two-flow convolutional neural network for visual tracking. *arXiv preprint arXiv:1604.07507*, 2016. (Cited on pages 69 and 88.)

[35] S. Chen. Kalman filter for robot vision: a survey. *IEEE Transactions on Industrial Electronics*, 2012. (Cited on page 14.)

[36] J. Choi, H. Jin Chang, J. Jeong, Y. Demiris, and J. Young Choi. Visual tracking using attention-modulated disintegration and integration. In *IEEE Conference on Computer Vision and Pattern Recognition*, 2016. (Cited on pages 78 and 79.)

[37] J. Choi, H. J. Chang, T. Fischer, S. Yun, K. Lee, J. Jeong, Y. Demiris, and J. Y. Choi. Context-aware deep feature compression for high-speed visual tracking. In *IEEE Conference on Computer Vision and Pattern Recognition*, 2018. (Cited on page 83.)

[38] S. Chopra, R. Hadsell, and Y. LeCun. Learning a similarity metric discriminatively, with application to face verification. In *IEEE Conference on Computer Vision and Pattern Recognition*, 2005. (Cited on pages 20, 26, and 118.)

[39] D. Comaniciu, V. Ramesh, and P. Meer. Real-time tracking of non-rigid objects using mean shift. In *IEEE Conference on Computer Vision and Pattern Recognition*. IEEE, 2000. (Cited on page 15.)

[40] G. Csurka. Domain adaptation for visual applications: A comprehensive survey. *arXiv preprint arXiv:1702.05374*, 2017. (Cited on page 119.)

[41] N. Dalal and B. Triggs. Histograms of oriented gradients for human detection. In *IEEE Conference on Computer Vision and Pattern Recognition*, 2005. (Cited on page 46.)

[42] M. Danelljan, G. Häger, F. S. Khan, and M. Felsberg. Accurate scale estimation for robust visual tracking. In *British Machine Vision Conference*, 2014. (Cited on pages 44, 46, 50, 56, 58, 60, 78, 89, 98, 99, 112, and 113.)

[43] M. Danelljan, F. S. Khan, M. Felsberg, and J. van de Weijer. Adaptive color attributes for real-time visual tracking. In *IEEE Conference on Computer Vision and Pattern Recognition*, 2014. (Cited on pages 8, 18, and 61.)

[44] M. Danelljan, G. Hager, F. Khan, and M. Felsberg. Convolutional features for correlation filter based visual tracking. In *IEEE International Conference on Computer Vision workshops*, 2015. (Cited on pages 18, 66, 69, 89, 99, 107, 109, and 114.)

[45] M. Danelljan, G. Häger, F. S. Khan, and M. Felsberg. Learning Spatially Regularized Correlation Filters for Visual Tracking. In *IEEE International Conference on Computer Vision*, 2015. (Cited on pages 18, 46, 58, 61, 77, 90, 98, and 99.)

[46] M. Danelljan, A. Robinson, F. S. Khan, and M. Felsberg. Beyond correlation filters: Learning continuous convolution operators for visual tracking. In *European Conference on Computer Vision*, 2016. (Cited on pages 83, 89, 90, 107, and 109.)

[47] M. Danelljan, G. Bhat, F. S. Khan, M. Felsberg, et al. Eco: Efficient convolution operators for tracking. In *IEEE Conference on Computer Vision and Pattern Recognition*, 2017. (Cited on pages 83 and 89.)

[48] K. De Brabanter and B. De Moor. Deconvolution in nonparametric statistics. In *ESANN*, 2012. (Cited on page 91.)

[49] M. Denil, L. Bazzani, H. Larochelle, and N. de Freitas. Learning where to attend with deep architectures for image tracking. *Neural Computation*, 2012. (Cited on page 68.)

[50] K. M. Dewar and F. Xu. Induction, overhypothesis, and the origin of abstract knowledge: Evidence from 9-month-old infants. *Psychological Science*, 2010. (Cited on page 5.)

[51] T. G. Dietterich, R. H. Lathrop, and T. Lozano-Pérez. Solving the multiple instance problem with axis-parallel rectangles. *Artificial intelligence*, 1997. (Cited on page 16.)

[52] X. Dong and J. Shen. Triplet loss in siamese network for object tracking. In *European Conference on Computer Vision*, 2018. (Cited on page 117.)

[53] A. Dosovitskiy, P. Fischer, E. Ilg, P. Hausser, C. Hazirbas, V. Golkov, P. van der Smagt, D. Cremers, and T. Brox. FlowNet: Learning optical flow with convolutional networks. In *IEEE International Conference on Computer Vision*, 2015. (Cited on page 71.)

[54] A. Doucet, N. De Freitas, and N. Gordon. An introduction to sequential monte carlo methods. In *Sequential Monte Carlo methods in practice*, pages 3–14. Springer, 2001. (Cited on page 18.)

[55] S. Duffner and C. Garcia. Pixeltrack: a fast adaptive algorithm for tracking non-rigid objects. In *IEEE International Conference on Computer Vision*, 2013. (Cited on pages 47 and 58.)

[56] T. Evgeniou and M. Pontil. Regularized multi–task learning. In *Proceedings of the tenth ACM SIGKDD international conference on Knowledge discovery and data mining*, 2004. (Cited on page 27.)

[57] H. Fan and H. Ling. Parallel tracking and verifying: A framework for real-time and high accuracy visual tracking. In *IEEE International Conference on Computer Vision*, 2017. (Cited on page 83.)

[58] H. Fan, L. Lin, F. Yang, P. Chu, G. Deng, S. Yu, H. Bai, Y. Xu, C. Liao, and H. Ling. Lasot: A high-quality benchmark for large-scale single object tracking. *arXiv preprint arXiv:1809.07845*, 2018. (Cited on pages 12, 13, 83, 84, 115, and 118.)

[59] L. Fei-Fei, R. Fergus, and P. Perona. One-shot learning of object categories. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2006. (Cited on pages 19 and 91.)

[60] P. F. Felzenszwalb, R. B. Girshick, D. McAllester, and D. Ramanan. Object detection with discriminatively trained part based models. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2010. (Cited on pages 8 and 17.)

[61] J. A. Fernandez and B. Vijayakumar. Zero-aliasing correlation filters. In *International Symposium on Image and Signal Processing and Analysis*, 2013. (Cited on pages 46 and 90.)

[62] M. Fink. Object classification from a single example utilizing class relevance metrics. In *Advances in Neural Information Processing Systems*, 2005. (Cited on page 20.)

[63] C. Finn and S. Levine. Meta-learning and universality: Deep representations and gradient descent can approximate any learning algorithm. In *International Conference on Learning Representations*, 2018. (Cited on page 23.)

[64] C. Finn, P. Abbeel, and S. Levine. Model-agnostic meta-learning for fast adaptation of deep networks. In *International Conference on Machine Learning*, 2017. (Cited on pages 23, 26, 27, 28, 36, 38, 39, 41, and 115.)

[65] C.-S. Fuh and P. Maragos. Motion displacement estimation using an affine model for image matching. *Optical Engineering*, 1991. (Cited on page 15.)

[66] K. Fukunaga and L. Hostetler. The estimation of the gradient of a density function, with applications in pattern recognition. *IEEE Transactions on information theory*, 1975. (Cited on page 15.)

[67] A. Gaidon, Q. Wang, Y. Cabon, and E. Vig. Virtual worlds as proxy for multi-object tracking analysis. In *IEEE Conference on Computer Vision and Pattern Recognition*, 2016. (Cited on page 3.)

[68] H. K. Galoogahi, A. Fagg, and S. Lucey. Learning background-aware correlation filters for visual tracking. In *IEEE International Conference on Computer Vision*, 2017. (Cited on page 83.)

[69] Q. Gan, Q. Guo, Z. Zhang, and K. Cho. First step toward model-free, anonymous object tracking with recurrent neural networks. *arXiv preprint arXiv:1511.06425*, 2015. (Cited on page 68.)

[70] J. Gao, H. Ling, W. Hu, and J. Xing. Transfer Learning Based Visual Tracking with Gaussian Processes Regression. In *European Conference on Computer Vision*, 2014. (Cited on page 60.)

[71] V. Garcia and J. Bruna. Few-shot learning with graph neural networks. In *International Conference on Learning Representations*, 2018. (Cited on pages 21, 37, 38, and 39.)

[72] S. Gidaris and N. Komodakis. Dynamic few-shot visual learning without forgetting. In *IEEE Conference on Computer Vision and Pattern Recognition*, 2018. (Cited on pages 20, 36, and 116.)

[73] M. Godec, P. M. Roth, and H. Bischof. Hough-based tracking of non-rigid objects. *Computer Vision and Image Understanding*, 2013. (Cited on page 47.)

[74] J. Goldberger, G. E. Hinton, S. T. Roweis, and R. R. Salakhutdinov. Neighbourhood components analysis. In *Advances in Neural Information Processing Systems*, 2005. (Cited on page 118.)

[75] B. Gong, Y. Shi, F. Sha, and K. Grauman. Geodesic flow kernel for unsupervised domain adaptation. In *IEEE Conference on Computer Vision and Pattern Recognition*, 2012. (Cited on page 27.)

[76] I. Goodfellow, Y. Bengio, A. Courville, and Y. Bengio. *Deep learning*. MIT press Cambridge, 2016. (Cited on pages 2 and 4.)

[77] N. Goodman. *Fact, fiction, and forecast*. Harvard University Press, 1983. (Cited on page 5.)

[78] H. Grabner and H. Bischof. On-line boosting and vision. In *IEEE Conference on Computer Vision and Pattern Recognition*, 2006. (Cited on page 16.)

[79] H. Grabner, C. Leistner, and H. Bischof. Semi-supervised on-line boosting for robust tracking. In *European Conference on Computer Vision*, 2008. (Cited on page 47.)

[80] H. Grabner, J. Matas, L. Van Gool, and P. Cattin. Tracking the invisible: Learning where the object might be. In *IEEE Conference on Computer Vision and Pattern Recognition*, 2010. (Cited on page 13.)

[81] A. Griewank and A. Walther. *Evaluating derivatives: Principles and techniques of algorithmic differentiation*. 2008. (Cited on page 96.)

[82] G. D. Hager and P. N. Belhumeur. Real-time tracking of image regions with changes in geometry and illumination. In *IEEE Conference on Computer Vision and Pattern Recognition*, 1996. (Cited on page 15.)

[83] S. Hare, A. Saffari, and P. H. S. Torr. Struck: Structured output tracking with kernels. In *IEEE International Conference on Computer Vision*, 2011. (Cited on pages 16, 43, 60, 66, 70, 77, and 79.)

[84] A. He, C. Luo, X. Tian, and W. Zeng. A twofold siamese network for real-time object tracking. In *IEEE Conference on Computer Vision and Pattern Recognition*, 2018. (Cited on page 117.)

[85] K. He, X. Zhang, S. Ren, and J. Sun. Delving deep into rectifiers: Surpassing human-level performance on ImageNet classification. In *IEEE International Conference on Computer Vision*, 2015. (Cited on page 77.)

[86] D. Held, S. Thrun, and S. Savarese. Learning to track at 100 fps with deep regression networks. In *European Conference on Computer Vision*, 2016. (Cited on pages 19, 69, 70, 88, and 90.)

[87] J. F. Henriques, R. Caseiro, P. Martins, and J. Batista. Exploiting the circulant structure of tracking-by-detection with kernels. In *European Conference on Computer Vision*, 2012. (Cited on page 17.)

[88] J. F. Henriques, R. Caseiro, P. Martins, and J. Batista. High-speed tracking with kernelized correlation filters. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2015. (Cited on pages 8, 17, 43, 45, 50, 56, 58, 66, 77, 79, 89, 98, 99, 100, 102, 103, and 124.)

[89] J. F. Henriques. *Circulant structures in computer vision*. PhD thesis, University of Coimbra, 2016. (Cited on page 54.)

[90] J. F. Henriques, J. Carreira, R. Caseiro, and J. Batista. Beyond hard negative mining: Efficient detector learning via block-circulant decomposition. In *IEEE International Conference on Computer Vision*, 2013. (Cited on page 45.)

[91] A. Hermans, L. Beyer, and B. Leibe. In defense of the triplet loss for person re-identification. *arXiv preprint arXiv:1703.07737*, 2017. (Cited on page 118.)

[92] C. F. Hester and D. Casasent. Multivariant technique for multiclass pattern recognition. *Applied Optics*, 1980. (Cited on page 17.)

[93] S. Hochreiter, A. S. Younger, and P. R. Conwell. Learning to learn using gradient descent. In *International Conference on Artificial Neural Networks*, pages 87–94. Springer, 2001. (Cited on pages 22 and 28.)

[94] S. Hong, T. You, S. Kwak, and B. Han. Online tracking by learning discriminative saliency map with convolutional neural network. In *International Conference on Machine Learning*, 2015. (Cited on pages 60 and 114.)

[95] Z. Hong, Z. Chen, C. Wang, X. Mei, D. Prokhorov, and D. Tao. Multi-store tracker (muster): a cognitive psychology inspired approach to object tracking. In *IEEE Conference on Computer Vision and Pattern Recognition*, 2015. (Cited on page 48.)

[96] D. Huang, L. Luo, M. Wen, Z. Chen, and C. Zhang. Enable scale and aspect ratio adaptability in visual tracking with detection proposals. 2015. (Cited on pages 78 and 79.)

[97] L. Huang, X. Zhao, and K. Huang. Got-10k: A large high-diversity benchmark for generic object tracking in the wild. *arXiv preprint arXiv:1810.11981*, 2018. (Cited on pages 12, 13, 83, 84, 115, and 118.)

[98] S. Ioffe and C. Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *International Conference on Machine Learning*, 2015. (Cited on page 76.)

[99] C. Ionescu, O. Vantzos, and C. Sminchisescu. Matrix backpropagation for deep networks with structured layers. In *IEEE International Conference on Computer Vision*, 2015. (Cited on pages 91 and 100.)

[100] S. E. Kahou, V. Michalski, and R. Memisevic. RATM: Recurrent Attentive Tracking Model. *arXiv preprint arXiv:1510.08660*, 2015. (Cited on page 68.)

[101] Ł. Kaiser, O. Nachum, A. Roy, and S. Bengio. Learning to remember rare events. In *International Conference on Learning Representations*, 2017. (Cited on pages 23 and 28.)

[102] Z. Kalal, K. Mikolajczyk, and J. Matas. Tracking-Learning-Detection. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2012. (Cited on pages 13, 16, 43, 47, 48, and 66.)

[103] W. Kay, J. Carreira, K. Simonyan, B. Zhang, C. Hillier, S. Vijayanarasimhan, F. Viola, T. Green, T. Back, P. Natsev, et al. The kinetics human action video dataset. *arXiv preprint arXiv:1705.06950*, 2017. (Cited on page 3.)

[104] H. Kiani Galoogahi, T. Sim, and S. Lucey. Multi-Channel Correlation Filters. In *IEEE International Conference on Computer Vision*, 2013. (Cited on pages 45 and 53.)

[105] H. Kiani Galoogahi, T. Sim, and S. Lucey. Correlation Filters with Limited Boundaries. In *IEEE Conference on Computer Vision and Pattern Recognition*, 2015. (Cited on pages 18, 46, 90, and 99.)

[106] D. P. Kingma and J. Ba. Adam: A method for stochastic optimization. 2015. (Cited on page 36.)

[107] G. Koch, R. Zemel, and R. Salakhutdinov. Siamese neural networks for one-shot image recognition. In *International Conference on Machine Learning workshops*, 2015. (Cited on pages 20, 26, 27, 38, and 71.)

[108] M. Kristan, R. Pflugfelder, A. Leonardis, J. Matas, F. Porikli, L. Cehovin, G. Nebehay, G. Fernandez, T. Vojir, A. Gatt, et al. The visual object tracking vot2013 challenge results. In *IEEE International Conference on Computer Vision workshops*, 2013. (Cited on pages 7 and 14.)

[109] M. Kristan, J. Matas, A. Leonardis, M. Felsberg, L. Cehovin, G. Fernández, T. Vojir, G. Hager, G. Nebehay, and R. Pflugfelder. The Visual Object Tracking VOT2014 challenge results. In *European Conference on Computer Vision workshops*, 2014. (Cited on pages 7, 14, 46, 56, 57, and 58.)

[110] M. Kristan, J. Matas, A. Leonardis, M. Felsberg, L. Cehovin, G. Fernandez, T. Vojir, G. Hager, G. Nebehay, R. Pflugfelder, et al. The Visual Object Tracking VOT2015 Challenge results. In *IEEE International Conference on Computer Vision workshops*, 2015. (Cited on pages 7, 14, 18, 56, 57, 59, and 75.)

[111] M. Kristan, A. Leonardis, J. Matas, M. Felsberg, R. Pflugfelder, L. Čehovin Zajc, T. Vojír, G. Häger, A. Lukežič, A. Eldesokey, G. Fernandez,

, et al. The Visual Object Tracking VOT challenge results. In *ICCV and ECCV workshops*, 2015, 2016, 2017, 2018.

[112] M. Kristan, A. Leonardis, J. Matas, M. Felsberg, R. Pflugfelder, L. Čehovin, T. Vojír, G. Häger, A. Lukežič, G. Fernández, et al. The Visual Object Tracking VOT2016 challenge results. In *European Conference on Computer Vision workshops*, 2016. (Cited on page 14.)

[113] M. Kristan, J. Matas, A. Leonardis, T. Vojíř, R. Pflugfelder, G. Fernandez, G. Nebehay, F. Porikli, and L. Čehovin. A novel performance evaluation methodology for single-target trackers. 2016. (Cited on pages 7, 12, 58, 67, 83, 103, 118, and 120.)

[114] M. Kristan, A. Leonardis, J. Matas, M. Felsberg, R. Pflugfelder, L. Čehovin Zajc, T. Vojir, G. Häger, A. Lukežič, A. Eldesokey, G. Fernandez, et al. The Visual Object Tracking VOT2017 challenge results. In *IEEE International Conference on Computer Vision workshops*, 2017. (Cited on pages 14, 18, 82, and 83.)

[115] M. Kristan, A. Leonardis, J. Matas, M. Felsberg, R. Pflugfelder, L. Čehovin Zajc, T. Vojir, G. Häger, A. Lukežič, A. Eldesokey, G. Fernandez, et al. The Visual Object Tracking VOT2018 challenge results. In *European Conference on Computer Vision workshops*, 2018. (Cited on pages 14 and 115.)

[116] A. Krizhevsky and G. Hinton. Learning multiple layers of features from tiny images. 2009. (Cited on pages 29 and 34.)

[117] A. Krizhevsky, I. Sutskever, and G. E. Hinton. Imagenet classification with deep convolutional neural networks. In *Advances in Neural Information Processing Systems*, 2012. (Cited on pages 18, 71, and 76.)

[118] B. V. Kumar, A. Mahalanobis, and R. D. Juday. *Correlation pattern recognition*. Cambridge University Press, 2005. (Cited on pages 17, 45, and 89.)

[119] A. Kuznetsova, H. Rom, N. Alldrin, J. Uijlings, I. Krasin, J. Pont-Tuset, S. Kamali, S. Popov, M. Malloci, T. Duerig, et al. The open images dataset v4: Unified image classification, object detection, and visual relationship detection at scale. *arXiv preprint arXiv:1811.00982*, 2018. (Cited on page 2.)

[120] J. Kwon and K. M. Lee. Visual Tracking Decomposition. In *IEEE Conference on Computer Vision and Pattern Recognition*, 2010. (Cited on page 48.)

[121] J. Kwon and K. M. Lee. Tracking by Sampling Trackers. In *IEEE International Conference on Computer Vision*, 2011. (Cited on page 48.)

[122] B. M. Lake, R. R. Salakhutdinov, and J. Tenenbaum. One-shot learning by inverting a compositional causal process. In *Advances in Neural Information Processing Systems*, 2013. (Cited on page 20.)

[123] B. M. Lake, R. Salakhutdinov, and J. B. Tenenbaum. Human-level concept learning through probabilistic program induction. *Science*, 2015. (Cited on pages 6, 20, 29, 33, 34, and 91.)

[124] L. Leal-Taixé, C. Canton-Ferrer, and K. Schindler. Learning by tracking: Siamese CNN for robust target association. In *IEEE Conference on Computer Vision and Pattern Recognition workshops*, 2016. (Cited on page 88.)

[125] Y. LeCun, Y. Bengio, and G. Hinton. Deep learning. *Nature*, 2015. (Cited on pages 3 and 18.)

[126] J.-Y. Lee and W. Yu. Visual tracking by partition-based histogram backprojection and maximum support criteria. In *Robotics and Biomimetics (ROBIO), 2011 IEEE International Conference on*. IEEE, 2011. (Cited on page 83.)

[127] B. Li, J. Yan, W. Wu, Z. Zhu, and X. Hu. High performance visual tracking with siamese region proposal network. In *IEEE Conference on Computer Vision and Pattern Recognition*, 2018. (Cited on pages 113 and 117.)

[128] Y. Li and J. Zhu. A scale adaptive kernel correlation filter tracker with feature integration. In *European Conference on Computer Vision workshops*, 2014. (Cited on pages 58, 89, 112, and 113.)

[129] P. Liang, E. Blasch, and H. Ling. Encoding color information for visual tracking: Algorithms and benchmark. In *IEEE Transactions on Image Processing*, 2015. (Cited on page 103.)

[130] M. W. Libbrecht and W. S. Noble. Machine learning applications in genetics and genomics. *Nature Reviews Genetics*, 2015. (Cited on page 1.)

[131] J. Long, E. Shelhamer, and T. Darrell. Fully convolutional networks for semantic segmentation. In *IEEE Conference on Computer Vision and Pattern Recognition*, 2015. (Cited on page 91.)

[132] B. D. Lucas, T. Kanade, et al. An iterative image registration technique with an application to stereo vision. In *IJCAI*, 1981. (Cited on page 15.)

[133] A. Lukežič, T. Vojíř, L. Čehovin, J. Matas, and M. Kristan. Discriminative correlation filter with channel and spatial reliability. In *IEEE Conference on Computer Vision and Pattern Recognition*, 2017. (Cited on pages 18, 83, and 90.)

[134] A. Lukežič, L. Č. Zajc, T. Vojíř, J. Matas, and M. Kristan. Now you see me: evaluating performance in long-term visual tracking. *arXiv preprint arXiv:1804.07056*, 2018. (Cited on pages 13, 48, 115, and 118.)

[135] W. Luo, A. G. Schwing, and R. Urtasun. Efficient deep learning for stereo matching. In *IEEE Conference on Computer Vision and Pattern Recognition*, 2016. (Cited on page 72.)

[136] C. Ma, J.-B. Huang, X. Yang, and M.-H. Yang. Hierarchical convolutional features for visual tracking. In *IEEE International Conference on Computer Vision*, 2015. (Cited on pages 66, 69, 89, 90, 99, and 107.)

[137] C. Ma, X. Yang, C. Zhang, and M.-H. Yang. Long-term Correlation Tracking. In *IEEE Conference on Computer Vision and Pattern Recognition*, 2015. (Cited on pages 13, 48, 78, 79, 89, 98, 99, 112, and 113.)

[138] M. L. Mack, B. C. Love, and A. R. Preston. Dynamic updating of hippocampal object representations reflects new conceptual knowledge. *Proceedings of the National Academy of Sciences*, 2016. (Cited on page 5.)

[139] D. Maclaurin, D. Duvenaud, and R. P. Adams. Gradient-based hyperparameter optimization through reversible learning. In *International Conference on Machine Learning*, 2015. (Cited on page 23.)

[140] A. Mahalanobis, B. V. Kumar, and D. Casasent. Minimum average correlation energy filters. *Applied Optics*, 1987. (Cited on page 17.)

[141] R. Manmatha and J. Oliensis. Extracting affine deformations from image patches-i: Finding scale and rotation. In *IEEE Conference on Computer Vision and Pattern Recognition*, 1993. (Cited on page 15.)

[142] M. McCloskey and N. J. Cohen. Catastrophic interference in connectionist networks: The sequential learning problem. In *Psychology of learning and motivation*. 1989. (Cited on pages 6 and 23.)

[143] A. Milan, L. Leal-Taixé, I. Reid, S. Roth, and K. Schindler. MOT16: A benchmark for multi-object tracking. *arXiv:1603.00831 [cs]*, 2016. `http://arxiv.org/abs/1603.00831`. (Cited on page 12.)

[144] E. G. Miller, N. E. Matsakis, and P. A. Viola. Learning from one example through shared densities on transforms. In *IEEE Conference on Computer Vision and Pattern Recognition*. IEEE, 2000. (Cited on page 19.)

[145] N. Mishra, M. Rohaninejad, X. Chen, and P. Abbeel. A simple neural attentive meta-learner. In *International Conference on Learning Representations*, 2018. (Cited on pages 23, 27, 35, and 38.)

[146] T. M. Mitchell. *The need for biases in learning generalizations*. Department of Computer Science, Laboratory for Computer Science Research, Rutgers Univ. New Jersey, 1980. (Cited on pages 21 and 24.)

[147] T. M. Mitchell et al. Machine learning. 1997. *Burr Ridge, IL: McGraw Hill*, 1997. (Cited on page 22.)

[148] A. Moudgil and V. Gandhi. Long-term visual object tracking benchmark. *arXiv preprint arXiv:1712.01358*, 2017. (Cited on pages 13, 48, 115, and 118.)

[149] Y. Movshovitz-Attias, A. Toshev, T. K. Leung, S. Ioffe, and S. Singh. No fuss distance metric learning using proxies. In *IEEE International Conference on Computer Vision*, 2017. (Cited on page 118.)

[150] M. Mueller, N. Smith, and B. Ghanem. A benchmark and simulator for uav tracking. In *European Conference on Computer Vision*, 2016. (Cited on page 67.)

[151] M. Müller, A. Bibi, S. Giancola, S. Al-Subaihi, and B. Ghanem. Trackingnet: A large-scale dataset and benchmark for object tracking in the wild. 2018. (Cited on pages 13, 115, and 118.)

[152] T. Munkhdalai and H. Yu. Meta networks. In *International Conference on Machine Learning*, 2017. (Cited on pages 22, 23, and 28.)

[153] K. P. Murphy. *Machine Learning: A Probabilistic Perspective*. The MIT Press, 2012. (Cited on pages 30 and 32.)

[154] I. Murray. Differentiation of the Cholesky decomposition. *arXiv preprint arXiv:1602.07527*, 2016. (Cited on page 91.)

[155] D. K. Naik and R. Mammone. Meta-neural networks that learn by learning. In *Neural Networks, 1992. IJCNN., International Joint Conference on*. IEEE, 1992. (Cited on page 21.)

[156] H. Nam and B. Han. Learning multi-domain convolutional neural networks for visual tracking. In *IEEE Conference on Computer Vision and Pattern Recognition*, 2016. (Cited on pages 18, 66, 69, 70, 77, 88, and 120.)

[157] A. Nichol, J. Achiam, and J. Schulman. On first-order meta-learning algorithms. *arXiv preprint arXiv:1803.02999*, 2018. (Cited on page 28.)

[158] J. Ning, J. Yang, S. Jiang, L. Zhang, and M.-H. Yang. Object tracking via dual linear structured svm and explicit feature map. In *IEEE Conference on Computer Vision and Pattern Recognition*, 2016. (Cited on page 78.)

[159] K. Nummiaro, E. Koller-Meier, and L. Van Gool. An adaptive color-based particle filter. *IVC*, 2003. (Cited on page 46.)

[160] S. W. Oh, J.-Y. Lee, K. Sunkavalli, and S. J. Kim. Fast video object segmentation by reference-guided mask propagation. In *IEEE Conference on Computer Vision and Pattern Recognition*, 2018. (Cited on page 119.)

[161] F. Orabona, C. Castellini, B. Caputo, A. E. Fiorilla, and G. Sandini. Model adaptation with least-squares svm for adaptive hand prosthetics. In *IEEE International Conference on Robotics and Automation*, 2009. (Cited on page 20.)

[162] N. C. Oza and S. Russell. *Online ensemble learning*. University of California, Berkeley, 2001. (Cited on page 16.)

[163] E. Park and A. C. Berg. Meta-tracker: Fast and robust online adaptation for visual object trackers. 2018. (Cited on page 120.)

[164] O. M. Parkhi, A. Vedaldi, and A. Zisserman. Deep face recognition. 2015. (Cited on page 71.)

[165] F. Perazzi. *Video Object Segmentation*. PhD thesis, ETH Zurich, 2017. (Cited on pages 12 and 119.)

[166] P. Pérez, C. Hue, J. Vermaak, and M. Gangnet. Color-Based Probabilistic Tracking. In *European Conference on Computer Vision*, 2002. (Cited on pages 15 and 46.)

[167] K. B. Petersen, M. S. Pedersen, et al. The matrix cookbook. *Technical University of Denmark*, 2008. (Cited on pages 10, 29, 30, and 116.)

[168] R. Pflugfelder. Siamese learning visual tracking: A survey. *arXiv preprint arXiv:1707.00569*, 2017. (Cited on page 19.)

[169] S. Pinker. *Language learnability and language development, with new commentary by the author*. Harvard University Press, 2009. (Cited on page 5.)

[170] J. Pont-Tuset, F. Perazzi, S. Caelles, P. Arbeláez, A. Sorkine-Hornung, and L. Van Gool. The 2017 davis challenge on video object segmentation. *arXiv preprint arXiv:1704.00675*, 2017. (Cited on page 12.)

[171] H. Possegger, T. Mauthner, and H. Bischof. In defense of color-based model-free tracking. In *IEEE Conference on Computer Vision and Pattern Recognition*, 2015. (Cited on pages 15, 44, 46, 55, and 58.)

[172] S. J. Prince. *Computer vision: models, learning, and inference*. Cambridge University Press, 2012. (Cited on page 1.)

[173] S. Qiao, C. Liu, W. Shen, and A. L. Yuille. Few-shot image recognition by predicting parameters from activations. In *IEEE Conference on Computer Vision and Pattern Recognition*, 2018. (Cited on page 36.)

[174] S. Ravi and H. Larochelle. Optimization as a model for few-shot learning. In *International Conference on Learning Representations*, 2017. (Cited on pages 4, 22, 27, 28, 34, 36, 38, and 115.)

[175] A. Razavian, H. Azizpour, J. Sullivan, and S. Carlsson. CNN features off-the-shelf: An astounding baseline for recognition. In *IEEE Conference on Computer Vision and Pattern Recognition workshops*, 2014. (Cited on page 71.)

[176] E. Real, J. Shlens, S. Mazzocchi, X. Pan, and V. Vanhoucke. Youtube-boundingboxes: A large high-precision human-annotated data set for object detection in video. In *IEEE Conference on Computer Vision and Pattern Recognition*, 2017. (Cited on pages 114, 117, and 120.)

[177] S.-A. Rebuffi, H. Bilen, and A. Vedaldi. Learning multiple visual domains with residual adapters. In *Advances in Neural Information Processing Systems*, 2017. (Cited on pages 22 and 119.)

[178] M. Ren, E. Triantafillou, S. Ravi, J. Snell, K. Swersky, J. B. Tenenbaum, H. Larochelle, and R. S. Zemel. Meta-learning for semi-supervised few-shot classification. In *International Conference on Learning Representations*, 2018. (Cited on page 26.)

[179] S. Ren, K. He, R. Girshick, and J. Sun. Faster r-cnn: Towards real-time object detection with region proposal networks. In *Advances in Neural Information Processing Systems*, 2015. (Cited on page 70.)

[180] S. R. Richter, Z. Hayder, and V. Koltun. Playing for benchmarks. In *IEEE International Conference on Computer Vision*, 2017. (Cited on page 3.)

[181] A. Rodriguez, V. N. Boddeti, B. V. K. V. Kumar, and A. Mahalanobis. Maximum margin correlation filter: A new approach for localization and classification. *IEEE Transactions on Image Processing*, 2013. (Cited on page 90.)

[182] D. A. Ross, J. Lim, R.-S. Lin, and M.-H. Yang. Incremental Learning for Robust Visual Tracking. *International Journal of Computer Vision*, 2008. (Cited on page 16.)

[183] S. Ruder. An overview of multi-task learning in deep neural networks. *arXiv preprint arXiv:1706.05098*, 2017. (Cited on page 27.)

[184] O. Russakovsky, J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma, Z. Huang, A. Karpathy, A. Khosla, M. Bernstein, A. C. Berg, and L. Fei-Fei. ImageNet Large Scale Visual Recognition Challenge. *International Journal of Computer Vision*, 2015. (Cited on pages 6, 67, 75, 85, and 86.)

[185] O. Russakovsky, J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma, Z. Huang, A. Karpathy, A. Khosla, M. Bernstein, et al. Imagenet large scale visual recognition challenge. 2015. (Cited on pages 34, 102, 114, 117, and 119.)

[186] S. J. Russell and P. Norvig. *Artificial intelligence: a modern approach*. Malaysia; Pearson Education Limited,, 2016. (Cited on page 1.)

[187] A. Saffari, C. Leistner, J. Santner, M. Godec, and H. Bischof. On-line random forests. In *IEEE International Conference on Computer Vision workshops*. IEEE, 2009. (Cited on page 16.)

[188] R. Salakhutdinov, J. Tenenbaum, and A. Torralba. One-shot learning with a hierarchical nonparametric bayesian model. In *Proceedings of ICML Workshop on Unsupervised and Transfer Learning*, 2012. (Cited on page 20.)

[189] J. Santner, C. Leistner, A. Saffari, T. Pock, and H. Bischof. Prost: Parallel robust online simple tracking. In *IEEE Conference on Computer Vision and Pattern Recognition*, 2010. (Cited on page 47.)

[190] A. Santoro, S. Bartunov, M. Botvinick, D. Wierstra, and T. Lillicrap. Meta-learning with memory-augmented neural networks. In *International Conference on Machine Learning*, 2016. (Cited on pages 23 and 28.)

[191] J. Schmidhuber. *Evolutionary principles in self-referential learning, or on learning how to learn: the meta-meta-... hook*. PhD thesis, Technische Universität München, 1987. (Cited on pages 9 and 21.)

[192] J. Schmidhuber. Learning to control fast-weight memories: An alternative to dynamic recurrent networks. *Neural Computation*, 1992. (Cited on page 22.)

[193] J. Schmidhuber. A neural network that embeds its own meta-levels. In *Neural Networks, 1993., IEEE International Conference on*. IEEE, 1993. (Cited on page 22.)

[194] F. Schroff, D. Kalenichenko, and J. Philbin. FaceNet: A unified embedding for face recognition and clustering. In *IEEE Conference on Computer Vision and Pattern Recognition*, 2015. (Cited on page 71.)

[195] P. Senna, I. N. Drummond, and G. S. Bastos. Real-time ensemble-based tracker with kalman filter. In *2017 30th SIBGRAPI Conference on Graphics, Patterns and Images (SIBGRAPI)*. IEEE, 2017. (Cited on page 83.)

[196] A. Sharif Razavian, H. Azizpour, J. Sullivan, and S. Carlsson. Cnn features off-the-shelf: an astounding baseline for recognition. In *IEEE Conference on Computer Vision and Pattern Recognition*, 2014. (Cited on page 107.)

[197] E. Simo-Serra, E. Trulls, L. Ferraz, I. Kokkinos, P. Fua, and F. Moreno-Noguer. Discriminative learning of deep convolutional feature point descriptors. In *IEEE International Conference on Computer Vision*, 2015. (Cited on page 71.)

[198] A. W. Smeulders, D. M. Chu, R. Cucchiara, S. Calderara, A. Dehghan, and M. Shah. Visual tracking: An experimental survey. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2014. (Cited on pages 7, 12, 14, 57, 66, 67, 75, and 118.)

[199] J. Snell, K. Swersky, and R. Zemel. Prototypical networks for few-shot learning. In *Advances in Neural Information Processing Systems*, 2017. (Cited on pages 20, 27, 35, 36, 38, 39, 41, 71, 91, and 116.)

[200] Y. Song, C. Ma, X. Wu, L. Gong, L. Bao, W. Zuo, C. Shen, R. Lau, and M.-H. Yang. Vital: Visual tracking via adversarial learning. In *IEEE Conference on Computer Vision and Pattern Recognition*, 2018. (Cited on page 18.)

[201] P. Sprechmann, S. M. Jayakumar, J. W. Rae, A. Pritzel, A. P. Badia, B. Uria, O. Vinyals, and D. Hassabis. Memory-based parameter adaptation. (Cited on page 23.)

[202] F. Sung, Y. Yang, L. Zhang, T. Xiang, P. H. S. Torr, and T. M. Hospedales. Learning to compare: Relation network for few-shot learning. In *IEEE Conference on Computer Vision and Pattern Recognition*, 2018. (Cited on pages 20, 37, 38, 39, 41, and 116.)

[203] J. S. Supančič and D. Ramanan. Self-paced learning for long-term tracking. In *IEEE Conference on Computer Vision and Pattern Recognition*, 2013. (Cited on pages 47 and 48.)

[204] Y. Taigman, M. Yang, M. Ranzato, and L. Wolf. DeepFace: Closing the gap to human-level performance in face verification. In *IEEE Conference on Computer Vision and Pattern Recognition*, 2014. (Cited on page 71.)

[205] F. Tang, S. Brennan, Q. Zhao, and H. Tao. Co-tracking using semi-supervised support vector machines. In *IEEE International Conference on Computer Vision*, 2007. (Cited on page 47.)

[206] R. Tao, E. Gavves, and A. W. Smeulders. Siamese instance search for tracking. In *IEEE Conference on Computer Vision and Pattern Recognition*, 2016. (Cited on pages 19, 69, 70, 77, 88, 90, and 113.)

[207] A. Tarantola. *Inverse problem theory and methods for model parameter estimation*, volume 89. siam, 2005. (Cited on page 121.)

[208] S. Thrun. Is learning the n-th thing any easier than learning the first? In *Advances in Neural Information Processing Systems*, 1996. (Cited on pages 5, 6, and 21.)

[209] S. Thrun and L. Pratt. *Learning to learn*. Springer Science & Business Media, 1998. (Cited on pages 21 and 27.)

[210] M. E. Tipping. The relevance vector machine. In *Advances in Neural Information Processing Systems*, 2000. (Cited on page 17.)

[211] T. Tommasi and B. Caputo. The more you know, the less you learn: from knowledge transfer to one-shot learning of object categories. In *British Machine Vision Conference*, 2009. (Cited on page 20.)

[212] P. H. S. Torr and A. Zisserman. Mlesac: A new robust estimator with application to estimating image geometry. *Computer vision and image understanding*, 2000. (Cited on page 48.)

[213] A. Torralba and A. A. Efros. Unbiased look at dataset bias. In *IEEE Conference on Computer Vision and Pattern Recognition*, 2011. (Cited on page 119.)

[214] E. Triantafillou, T. Zhu, V. Dumoulin, P. Lamblin, K. Xu, R. Goroshin, C. Gelada, K. Swersky, P.-A. Manzagol, and H. Larochelle. Meta-dataset: A dataset of datasets for learning to learn from few examples. (Cited on page 119.)

[215] P. E. Utgoff. Shift of bias for inductive concept learning. *Machine learning: An artificial intelligence approach*, 1986. (Cited on page 21.)

[216] J. Valmadre. *Stationary processes for object detection and non-rigid structure-from-motion*. PhD thesis, Queensland University of Technology, 2016. (Cited on page 53.)

[217] J. Valmadre, S. Sridharan, and S. Lucey. Learning Detectors Quickly with Stationary Statistics. 2014. (Cited on page 99.)

[218] J. Valmadre*, L. Bertinetto*, J. F. Henriques, A. Vedaldi, and P. H. S. Torr. End-to-end representation learning for correlation filter based tracking. In *IEEE Conference on Computer Vision and Pattern Recognition*, 2017. (Cited on page 11.)

[219] J. Valmadre*, L. Bertinetto*, J. F. Henriques, R. Tao, A. Vedaldi, A. Smeulders, P. H. S. Torr, and E. Gavves*. Long-term tracking in the wild: A benchmark. In *European Conference on Computer Vision*, 2018. (Cited on pages 12, 13, 48, 115, and 118.)

[220] R. Vilalta and Y. Drissi. A perspective view and survey of meta-learning. *Artificial Intelligence Review*, 2002. (Cited on pages 5, 6, 21, 22, 24, 91, and 116.)

[221] P. Vincent, H. Larochelle, I. Lajoie, Y. Bengio, and P.-A. Manzagol. Stacked denoising autoencoders: Learning useful representations in a deep network with a local denoising criterion. *Journal of machine learning research*, 2010. (Cited on page 18.)

[222] O. Vinyals, C. Blundell, T. Lillicrap, D. Wierstra, et al. Matching networks for one shot learning. In *Advances in Neural Information Processing Systems*, 2016. (Cited on pages 4, 6, 20, 23, 24, 27, 28, 29, 33, 34, 36, 38, 71, 91, and 115.)

[223] T. Vojir, J. Noskova, and J. Matas. Robust scale-adaptive mean-shift for tracking. *Pattern Recognition Letters*, 2014. (Cited on page 81.)

[224] L. Wang, W. Ouyang, X. Wang, and H. Lu. Visual tracking with fully convolutional networks. In *IEEE International Conference on Computer Vision*, 2015. (Cited on pages 18, 66, and 69.)

[225] N. Wang and D.-Y. Yeung. Learning a deep compact image representation for visual tracking. In *Advances in Neural Information Processing Systems*, 2013. (Cited on page 18.)

[226] N. Wang and D.-Y. Yeung. Ensemble-based tracking: Aggregating crowdsourced structured time series data. In *International Conference on Machine Learning*, 2014. (Cited on pages 48 and 60.)

[227] N. Wang, S. Li, A. Gupta, and D.-Y. Yeung. Transferring rich feature hierarchies for robust visual tracking. *arXiv preprint arXiv:1501.04587*, 2015. (Cited on pages 18, 61, 66, 69, 88, 89, and 114.)

[228] Q. Wang*, L. Zhang*, L. Bertinetto*, W. Hu, and P. H. S. Torr. Fast online object tracking and segmentation: A unifying approach. In *IEEE Conference on Computer Vision and Pattern Recognition*, 2019. (Cited on pages 12 and 119.)

[229] X. Wang and A. Gupta. Unsupervised learning of visual representations using videos. In *IEEE International Conference on Computer Vision*, 2015. (Cited on page 7.)

[230] O. Williams, A. Blake, and R. Cipolla. Sparse bayesian learning for efficient visual tracking. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2005. (Cited on pages 13 and 17.)

[231] M. Wrenninge and U. Jonas. Synscapes: A photorealistic synthetic dataset for street scene parsing. *arXiv:1810.08705 [cs]*, 2018. `http://arxiv.org/abs/1810.08705`. (Cited on page 3.)

[232] Y. Wu, J. Lim, and M.-H. Yang. Online object tracking: A benchmark. In *IEEE Conference on Computer Vision and Pattern Recognition*, 2013. (Cited on pages 12, 56, 60, 62, 67, 75, 78, 80, 83, 84, 103, and 111.)

[233] Y. Wu, J. Lim, and M.-H. Yang. Object tracking benchmark. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2015. (Cited on pages 7, 13, 14, 83, 103, 113, 118, and 120.)

[234] Y. Wu and K. He. Group normalization. *arXiv preprint arXiv:1803.08494*, 2018. (Cited on page 39.)

[235] J. Xiao, R. Stolkin, and A. Leonardis. Single target tracking using adaptive clustered decision trees and dynamic multi-level appearance models. In *IEEE Conference on Computer Vision and Pattern Recognition*, 2015. (Cited on pages 47 and 58.)

[236] N. Xu, L. Yang, Y. Fan, J. Yang, D. Yue, Y. Liang, B. Price, S. Cohen, and T. Huang. Youtube-vos: Sequence-to-sequence video object segmentation. In *European Conference on Computer Vision*, 2018. (Cited on pages 117 and 120.)

[237] L. Yang, Y. Wang, X. Xiong, J. Yang, and A. K. Katsaggelos. Efficient video object segmentation via network modulation. In *IEEE Conference on Computer Vision and Pattern Recognition*, 2018. (Cited on page 119.)

[238] T. Yang and A. B. Chan. Learning dynamic memory networks for object tracking. 2018. (Cited on pages 77 and 117.)

[239] A. Yilmaz, O. Javed, and M. Shah. Object tracking: A survey. *Acm computing surveys (CSUR)*, 2006. (Cited on pages 7 and 14.)

[240] A. S. Younger, S. Hochreiter, and P. R. Conwell. Meta-learning with backpropagation. In *Neural Networks, 2001. Proceedings. IJCNN'01. International Joint Conference on*. IEEE, 2001. (Cited on page 22.)

[241] S. Zagoruyko and N. Komodakis. Learning to compare image patches via convolutional neural networks. In *IEEE Conference on Computer Vision and Pattern Recognition*, 2015. (Cited on page 71.)

[242] M. D. Zeiler, D. Krishnan, G. W. Taylor, and R. Fergus. Deconvolutional networks. In *IEEE Conference on Computer Vision and Pattern Recognition*, 2010. (Cited on page 91.)

[243] M. Zhai, M. J. Roshtkhari, and G. Mori. Deep learning of appearance models for online object tracking. *arXiv preprint arXiv:1607.02568*, 2016. (Cited on page 88.)

[244] J. Zhang, S. Ma, and S. Sclaroff. Meem: Robust tracking via multiple experts using entropy minimization. In *European Conference on Computer Vision*, 2014. (Cited on pages 48 and 60.)

[245] Y. Zhang, L. Wang, J. Qi, D. Wang, M. Feng, and H. Lu. Structured siamese network for real-time visual tracking. In *European Conference on Computer Vision*, 2018. (Cited on pages 83 and 117.)

[246] G. Zhu, J. Wang, Y. Wu, and H. Lu. Collaborative correlation tracking. In *British Machine Vision Conference*, 2015. (Cited on page 78.)

[247] Z. Zhu, Q. Wang, B. Li, W. Wu, J. Yan, and W. Hu. Distractor-aware siamese networks for visual object tracking. In *European Conference on Computer Vision*, 2018. (Cited on pages 113 and 117.)

[248] K. Zimmermann, J. Matas, and T. Svoboda. Tracking by an optimal sequence of linear predictors. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2009. (Cited on page 17.)

[249] C. L. Zitnick and P. Dollár. Edge boxes: Locating object proposals from edges. In *European Conference on Computer Vision*, 2014. (Cited on page 79.)