Knowledge Distillation for Discriminative Correlation Filter Trackers

Student: Duolikun Danier

Supervisor: Gabriel J. Brostow

MSc Machine Learning

August 2020

Department of Computer Science

University College London

Abstract

Visual object tracking, where the task is to locate a pre-defined target in a sequence of video frames, has long been one of the main challenges in computer vision with a wide range of applications in areas such as surveillance, filming and robotics. While deep learning algorithms, especially convolutional neural networks (CNNs), have empowered modern trackers to achieve higher accuracy and robustness, they have also caused heavier computational burden which prohibits the deployment of state-of-the-art discriminative correlation filter-based trackers in many embedded platforms, e.g. mobile phones, where powerful computational resources such as GPUs are absent. On the other hand, it has been widely shown that it is difficult to train fast and small neural network with competitive performance. Therefore, there is a need for a solution with which one can obtain light-weighted trackers that can run real-time on CPUs while tracking accurately and reliably.

In this project, we apply knowledge distillation methods to visual tracking and propose a novel method called Correlation Filter Knowledge Distillation (CFKD) to transfer the knowledge of a sophisticated deep tracker (the teacher) to a more light-weighted one (the student) that can run in real time on a single-core CPU. Our developed method is specialised for the ATOM [19] family of trackers which dominate in the VOT-2019 [50] visual tracking competition. It combines a selection of loss functions including a novel correlation filter loss and offline distils the heavy teacher network in a multi-task learning approach. Different from previous work where the small student architecture is created by pruning the teacher network, we benefit from state-of-the-art work on developing CPU-efficient CNN architectures and deploy the MobileNetV2 [83] as the backbone of our student network.

We carry out extensive experiments to study the our CFKD method and our results show that it outperforms existing methods for distilling trackers in terms of tracking performance of the student. The evaluation on several popular tracking benchmarks show that with CFKD, one can train an ATOM-based student network that can run in real time on CPUs (at more than 20 frames-per-second) whilst retaining state-of-the-art tracking performance. Our code and results are available at `https://github.com/danielism97/CFKD`.

Acknowledgements

I would like to thank my parents for raising me with great respect, love and patience and supporting all my pursuits mentally and financially. Next, I would like to thank my supervisor, Professor Gabriel Brostow, for providing me with the interesting topic of research and supporting me academically via regular meetings thoughout the project period. Thirdly, I thank UCL Prism research group for generously providing constructive suggestions on my research direction and many stimulating thoughts. I would also like to acknowledge that the code is developed in this project under the PyTracking framework provided by Danelljan et al..

Contents

List of Figures

List of Tables

Chapter 1

Introduction and Background

Visual object tracking is one of the major challenges in computer vision. While various algorithms have been proposed to improve tracking accuracy and robustness, this project focuses on applying knowledge distillation methods to state-of-the-art trackers so as to improve the tracking speed. In the beginning chapter we introduce the problem to be addressed in this project, including its importance, barriers as well as relevance to other areas. A summary of our work and contributions are provided, followed by statement of our goals. In Section 1.4 and 1.5, we introduce necessary background knowledge. Finally, the overall structure of this report will be given.

1.1 Motivation

Visual object tracking, where the task is to locate a pre-defined target in a sequence of frames, has long been one of the major challenges in computer vision. It can be applied in many areas such as surveillance, filming, robotics and autonomous driving, just to name a few. In most of these scenarios, an ideal visual tracker should report the location of the target accurately with as few instances of failure as possible in real time. Therefore, *accuracy*, *robustness* and *speed* are three major factors in determining tracking performance.

Advancement of Deep Learning algorithms in the past decade has enabled significant improvement of models in many problems in computer vision, e.g. image classification [85] [36], object detection [30] [75], semantic segmentation [35] [78], etc. With deep neural networks allowing models in some of the fields to achieve human-level performance, deep Convolutional Neural Networks (CNNs) has also been widely deployed in visual object tracking due to their strong representational power. Reported in VOT-2019[1] [50] challenge, all of the best-performing trackers involve deep CNNs as part of them. Specifically, 7 of the top 10 trackers in the VOT-2019 Short Term Challenge[2] are based on Discriminative Correlation Filters (DCF)

[1]One of the main tracking benchmarks updated annually.

[2]Part of the VOT-2019 challenge in which trackers are evaluated on short videos, without requirement for speed.

and involve deep CNNs as feature extraction backbones. While these DCF trackers achieve state-of-the-art accuracy and robustness, they suffer from the heavy computational burden introduced by these deep backbones limiting their real-time performance on GPUs (graphics processing units), as indicated by the results of VOT-2019 Real-Time challenge[3] in which trackers are tested at 20 FPS (frames per second).

In addition, even though some of the deep DCF trackers can achieve real-time tracking on GPUs, due to their large size and computational cost, it is not feasible to deploy them on many embedded platforms, e.g. mobile phones, UAVs (unarmed aerial vehicles), which also require real-time tracking but have limiting budget on power, memory and computation. Therefore, there is a need for a solution that accelerates these deep trackers on low-budget devices while maintaining state-of-the-art tracking accuracy and robustness.

Knowledge Distillation methods [38] [12] are designed to train small "student" networks to achieve similar or better performance than large "teacher" networks by transferring the knowledge of the "teacher" to the "student". This is done by devising appropriate loss functions to force the "student" to mimic the output of the "teacher". While such techniques have been widely investigated in many other areas, work on applying knowledge distillation to visual tracking remains scarce, and the existing methods [103][94][63] only focus on specific families of trackers so might not generalise well to the state-of-the-art tracking framework ATOM (Accurate Tracking via Overlap Maximisation) [19]. ATOM deploys a CNN formulation of discriminative correlation filter to predict a rough position of the target then estimates the target bounding box with an IoUNet introduced in [43]. ATOM presented a novel framework for DCF trackers and enhanced their overall performance, indicated by the fact that all of the top 7 trackers in VOT-2019 Short Term Challenge are based on ATOM, and they achieved better scores as compared to VOT-2018 [48] (which share almost the same test dataset as VOT-2019). **Therefore in this project, we investigate the efficacy of applying knowledge distillation to visual tracking in order to create strong visual trackers that can run real-time on embedded platforms. In specific, we focus on distilling the ATOM family of trackers and propose a knowledge distillation method that combines novel loss functions and training strategy for distilling the knowledge of the off-the-shelf ATOM trackers with deep feature extraction backbones to smaller models that can run in real time on a single-core 2.3GHz Intel Xeon CPU achieving state-of-the-art tracking accuracy and robustness**.

Generally speaking, applying knowledge distillation to visual tracking can be hard because of the following reasons:

[3]Part of the VOT-2019 challenge that evaluates trackers at 20 frames per second.

1. There exist various DCF trackers. While some use deep CNNs as feature extractors, some may use them as an end-to-end network; while some use CNNs to predict bounding box coordinates, some use them to generate confidence maps. Therefore, it is difficult to develop a generic knowledge distillation strategy that fits all trackers.

2. For DCF trackers that are designed to train discriminative classifiers online, online distillation can add additional cost that may not be compensated by the improvements brought.

3. Trackers have to deal with various scenarios so need strong representation capabilities, but the small networks in distillation might not have the capacity, or it can be difficult to devise loss functions for exploiting their capacities.

To address these issues, we only focus on the state-of-the-art ATOM family of trackers, and develop a set of loss functions to perform the whole distillation process offline. Alongside transferring the semantic representational power of the teacher feature extractor to the student network using a fidelity loss, we also devise a correlation filter loss to inject tracking-specific knowledge so as to aid the distillation process. The previous work designs the student network architecture by reducing the teacher network layer-wise while maintaining the overall architecture. In contrast, we benefit from the existing work on CNN architectures and deploy the MobileNetV2 [83], which is designed to infer efficiently on CPUs, as the student network. In order to validate our method, we evaluate our proposed method as well as existing distillation methods on ATOM and provide quantitative comparison between them. The distilled ATOM trackers are also evaluated on popular tracking benchmarks including OTB-100 [98], LaSOT [28], TrackingNet [70], VOT-2018 [48] and VOT-2019 [50]. The results show that the ATOM trackers distilled with our method can achieve state-of-the-art tracking performance on these benchmarks while requiring much less memory and computation than other competing trackers, managing to track real-time on a single core CPU. Finally to verify the generalisation ability of our method on other ATOM trackers, we evaluate it to DRNet, the winner of the VOT-2019 challenge.

1.2 Goals

The goal of this project is to develop an effective knowledge distillation method to distil the state-of-the-art discriminative correlation filter-based trackers under the ATOM [19] framework, i.e. to design loss functions and training strategy to transfer the knowledge of a accurate yet slow tracker to a compact tracker that is expected to achieve similar performance to the large model. Our hypothesis is that with appropriate loss functions and distillation strategy, one can use a heavy and accurate tracker to train a smaller one that can achieve similar tracking performance as the heavy one. Studying the ATOM tracking algorithm, we will devise loss functions to distil different parts of the tracker. Although there has been several attempts to apply knowledge distillation in visual tracking where there is a high demand for compact models, due to the variety of tracking algorithms, the coverage of existing work is still narrow, leaving a large research gap between tracking and knowledge distillation. Therefore, we aim to fill part of the gap by developing a distillation method for the emerging family of ATOM trackers that define the state-of-the-art. Quantitatively, the desired method should build a light-weighed tracking model that can run in real time on CPUs and see minimal drop in tracking performance, and such drop should be compensated by the improvement in tracking speed. Therefore, instead of aiming for a compact tracker that can achieve higher scores on tracking benchmarks, our focus is on the distillation framework for ATOM trackers. Finally, although we focus on ATOM family, the loss functions and training strategy we proposed may benefit the area by providing elements of distillation methods for other trackers.

1.3 Summary of Work and Contributions

As will be introduced in detail in Chapter 3, ATOM comprises three parts: feature extraction backbone, the IoUNet (can be seen as the bounding box regressor), and the classifier. The classifier is a two-layer CNN trained efficiently online to learn specific appearance features of the target, so it is not concerned in our distillation method. To distil the teacher backbone, we use the fidelity loss to force the student backbone to mimic the responses of the teacher. In addition, we also use a novel correlation filter loss to guide the student to learn discriminative features for the targets. For the IoUNet, we train the student with the IoU loss that makes use of both the hard groundtruth labels and the soft labels produced by the teacher. We perform the whole process offline in a multi-task learning manner. In contrast to previous works that design the student architecture manually, we deploy the MobileNetV2 [83] designed for mobile platforms as the student backbone. Our distilled ATOM tracker has model size of 18 MB while the teacher model takes 109MB. The student can run on at 20 FPS (deemed real-time) on a single-core 2.3GHz Intel Xeon CPU processor compared to 9 FPS of the teacher (more than 200% increase), with performance drop of around 6% on the benchmarks.

We summarise our contributions as follows:

1. We propose a novel distillation method for the state-of-the-art ATOM family of trackers. In specific, on top of the fidelity loss, we devise a novel correlation filter loss to aid the distillation process and IoU losses to distil the bounding box regressor.

2. We propose to perform the distillation process in a multi-task manner such that the errors in the bounding box regressor outputs are allowed to flow into the feature extraction backbone during back-propagation. This has been shown to enhance the performance of the distilled model.

3. We implement and evaluate the existing methods for distilling trackers on ATOM and show that our method outperforms all existing methods.

4. We evaluate our method on the winner of the VOT-2019 challenge, DRNet, and show that the distilled DRNet achieves great improvement in tracking speed with very small drop in performance, indicating that our method is applicable for the general ATOM family.

In the rest of this chapter, we familiarise the reader with the necessary background knowledge in visual object tracking (Section 1.4) and knowledge distillation (Section 1.5), introducing key concepts in both areas.

1.4 Visual Object Tracking

In this section we review the definition and relevance of visual object tracking and introduce some of the key concepts in tracking such as appearance modelling. Then we introduce the two main categories into which modern deep-learning-based state-of-the-art trackers have developed, together with their relevance to our work.

In visual object tracking, the task is to infer the position and scale of the target in a sequence of frames given its location in the first frame. From a probabilistic point of view, it can be defined as a state estimation problem in which the state is the actual location of the target in the 2D frame and the observations are the pixel values in the frame. With the advancement of computing power over the past few decades, visual tracking has been widely applied to many fields. For example, a tracking algorithm that combines an object detection model and feature matching methods is developed in [27] for home robots. In [1], an automated surveillance system is realised by integrating motion detection and visual tracking.

In these tracking scenarios, the tracking algorithms should ideally be able to distinguish the target from the background, predict or detect the motion of the target, and report the location of the target in real time. Performing these tasks for 3D objects using only their 2D projection onto images brings about unique challenges in object tracking, including variations in object appearance, abrupt changes in object or camera motion, occlusions and varying background, etc.

In order to tackle these challenges, tracking algorithms have been developed generally to contain two main components: the appearance model and the motion model. The motion model is responsible for predicting the possible state (i.e. position and scale) of the target in the next frame, and popular methods include Kalman filter [16] and particle filter [60]. The appearance model aims to (i) develop visual representation of the target using effective feature descriptors, and (ii) use statistical modelling methods to learn mathematical models for identifying the object [82]. While one may argue that these two components are equally important, authors of [93] performed ablative experiments to study the effects of different components on tracking performance, and showed that the appearance model plays a much more important role than the motion model.

Conventional visual representation methods make use of raw pixel values, optical flow information, or hand-crafted local features such as HoG [10], SIFT [102], etc. However, with the empirical evidence indicating that deep neural networks work as robust feature extractors, more and more trackers deploy CNNs to build visual representation. It is noticed that in the VOT-2019 tracking challenge, all of the well-performing trackers adopt CNNs for appearance modelling. Despite such commonness in visual representation, the state-of-the-art trackers have diverged in their statistical modelling methods into two classes: generative models and discriminative models. Generative models concern the data generation process and adaptively generate and update an appearance template of the target with which the region in a new frame that is most similar to the target is searched for. State-of-the-art trackers that fall into this category often deploy a Siamese architecture, in which a CNN learns a similarity function to match images. On the other hand, discriminative models formulate tracking as a binary classification problem, in which the goal is to discriminate the target from the background. This mechanism is also called tracking-by-detection since the target is to be detected in each frame. Such trackers tend to train a discriminative correlation filter (DCF) offline and directly search for the target in a frame without needing to matching the image with a template. In the following, these two types of trackers are introduced in more details, including their working principles, advantages and disadvantages, as well as some of the typical models in each category.

1.4.1 Generative Siamese Trackers

As the name of the category suggests, this type of trackers typically deploy a Siamese architecture, which means that the target image and the search image are both passed through a backbone CNN to obtain their respective feature representations. This feature extraction step is typically followed by a cross-correlation operation between the feature maps of the search image and that of the target image. In the 2D spatial domain of images, the cross-correlation operation between two signals (images) measures similarity between them. Concretely, given two images or feature maps f, g with the same number of channels, the operation $f * g$ outputs a 2D map that has the highest value at the position corresponding to the region in f that is most similar to g. As such, the inference in Siamese trackers are generally done by finding the region in the search image that corresponds to the peak value in the cross-correlation output.

One of the typical trackers in this category is SiamFC proposed by Bertinetto et al. [8] and its architecture is shown in Figure 1.1, where it can be seen that the target image of the eagle and the search image are fed into the CNN backbone to obtain a feature map, which is then cross-correlated to obtain a score map. Although state-of-the-art Siamese trackers (e.g. SiamRPN [56], SiamRPN++ [54], SiamMask [95]) differ in the specific details of their tracking pipeline, they share the basic overall idea of Siamese networks and cross-correlation as described above.

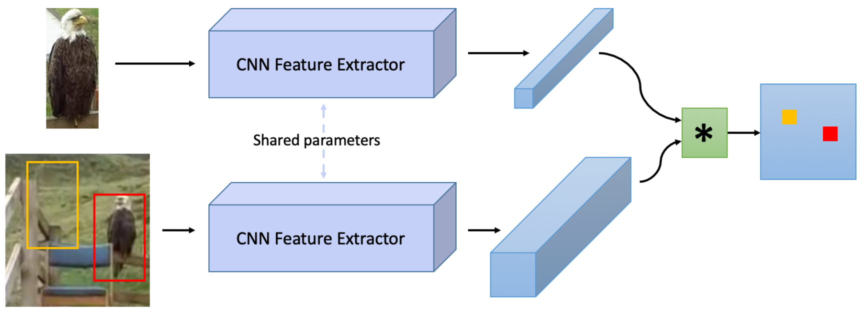

Figure 1.1: Architecture of SiamFC [8], where $*$ denotes cross-correlation.

One of the advantages of Siamese trackers is that they have little or no reliance on the motion model. The convolutional feature extraction and matching manner of Siamese networks allows a search image of arbitrary size to be processed in one pass, without the need for a motion model to predict a coarse location of the target and cropping of the search image according to such prediction. As a result, any potential failures due to inaccurate motion model can be avoided. In addition, Siamese networks are typically trained extensively offline to learn general similarity functions, thus requiring little online training that can potentially harm the tracking speed.

While these advantages of Siamese trackers allow them to dominate in the competition of real-time trackers (VOT-2019 Real-time Challenge reports that 7 of the top 10 trackers are Siamese-based), they come behind discriminative methods in terms of accuracy and robustness when speed is not considered, as shown by the results of VOT-2019 Short-term challenge. One obvious disadvantage of Siamese trackers is the restriction on their feature extractor caused by their working mechanism: during tracking, both the (adaptively updated) target template and the search image need to pass through the backbone, and such two-pass manner imposes restrictions on computational budget of the feature extractor. Empirical evidence show that most real-time Siamese trackers deploy shallower networks (e.g. AlexNet [51], VGG-M [85]) as backbones, while discriminative methods tend to benefit from the richer representations obtained from deeper networks such as the ResNet [36] family. Besides the lack of rich feature representations, Siamese trackers do not concern any classification but only matching of the template and because they are not trained online to adapt to appearance changes, they appear sensitive to distractors.

1.4.2 Discriminative Correlation Filter Trackers

Discriminative correlation filter (DCF) trackers have been widely studied and developed since Bolme et al. [11] proposed the Minimum Output Sum of Squared Error (MOSSE) framework for training DCFs. Under this framework, a correlation filter $\mathbf{w}$ can be trained using a single image $\mathbf{x}$ with the objective

$$\min_{\mathbf{w}} \| \sum_{i=1}^{C} \varphi_i(\mathbf{x}) \star \mathbf{w}_i - \mathbf{g} \|^2 \tag{1.1}$$

where $\varphi(\cdot)$ represents feature representation of the image and the index $i \in \{1, \ldots, C\}$ indicates the channel of the feature map. The $\star$ symbol denotes circular correlation operation, which is basically cross-correlation operation with one signal padded in a circular pattern, and $\mathbf{g}$ is the groundtruth 2D label representing a Gaussian function centred at the target location (although in the original formulation [11] $\mathbf{g}$ is a 2D binary mask for classification.). The intuition of such formulation is that when the feature representation of the search image is cross-correlated with the filter $\mathbf{w}$, the output should indicate the position of the target. As such, there is no need for matching the template as in the case of generative methods. Making use of the properties of Fourier Transform, one can train the DCF efficiently in the Fourier domain with Equation 1.2 [11]:

$$\hat{\mathbf{w}}_i = \frac{\hat{\mathbf{g}}^* \odot \hat{\varphi}_i(\mathbf{x})}{\sum_{j=1}^{C} \hat{\varphi}_j(\mathbf{x}) \odot \hat{\varphi}_j^*(\mathbf{x})} \tag{1.2}$$

where $\hat{\cdot}$ denotes Discrete Fourier Transform, $\cdot^*$ means complex conjugate, $\odot$ stands for element-wise multiplication. While traditional DCF trackers and their variations are mostly based on hand-crafted features, development of CNNs has shifted DCF trackers towards using features extracted from CNNs and formulating DCF in terms of CNNs. CNN formulation of DCF means that the correlation filter $\mathbf{w}$ and the circular correlation operation are replaced by a single CNN denoted by $f_\theta(\cdot)$ where θ is the parameters in the CNN. The objective now becomes

$$\min_{\theta} \| f_\theta(\varphi(\mathbf{x})) - \mathbf{g} \|^2 \tag{1.3}$$

and the DCF (parameters θ) can be trained by optimising this objective.

One of the representative trackers from this category is MDnet (Multi-Domain network) proposed by Nam et al. [72]. The architecture of MDnet is shown in Figure 1.2, where it can be seen that the tracker only has an end-to-end CNN that takes a search image patch as input and directly outputs the probability of the target being at the centre of the patch. MDnet pre-trains the convolutional layers offline on tracking datasets to learn generic feature representations. In tracking phase, the convolutional layers are frozen and the fully connected layers, which can be seen as neural network formulation of DCF, are trained online to learn domain-specific representations.

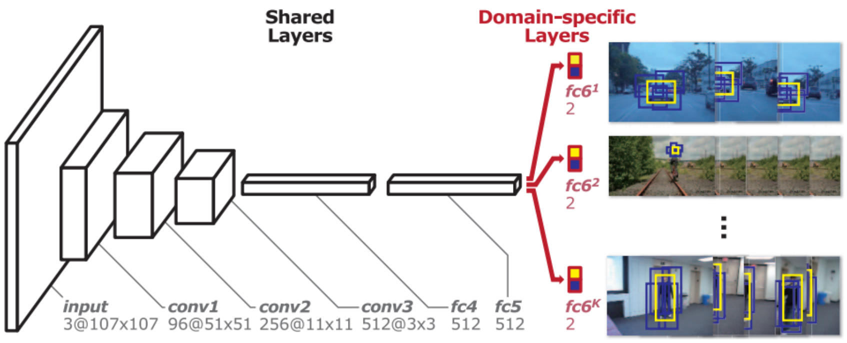

Figure 1.2: Architecture of MDnet [72]. Image is from the original paper.

Since tracking is posed as a binary classification problem in discriminative methods, trackers are trained explicitly to separate the target from the background. One may argue that such formulation is more suitable to tackle the challenges in object tracking such as occlusions and variations in the background. Indeed, experiments in [58] show that DCF trackers are more robust to distractors. Furthermore, to achieve better classification, most discriminative trackers make use of the rich and informative feature representations provided by deep CNNs. As a result, they can benefit from the advanced architectures developed for object classification problem. These advantages have allowed DCF trackers to achieve higher accuracy and robustness, as indicated by recent VOT Short-term challenges [48][50].

However, DCF trackers also have disadvantages that limit their real-time performance. Firstly, their feature extractors, e.g. ResNet family [36], VGG nets [85], are deep and heavy, which increases the computational burden of the trackers and makes their deployment on lots of embedded platforms infeasible. Secondly, most DCF trackers apply online training strategy to their correlation filters which introduces even more computational cost that makes them slower than Siamese trackers.

The introduction of ATOM (Accurate Tracking by Overlap Maximisation) tracker by

Danelljan et al. [19] has brought about great improvements in tracking performance. Reported in [50], 7 of the top 10 trackers in the VOT-2019 Short-term challenge are based on ATOM. Besides the innovation in bounding-box regression method, ATOM proposed to use more efficient optimisation methods such as Gauss-Newton for online-training of the DCF, which significantly alleviated the second problem of DCF trackers described above. However, the backbone architecture of these DCF trackers (mostly ResNet-18) is still too heavy to be deployed on CPUs. While the tracking mechanism of DCF trackers has enabled their high accuracy and robustness than Siamese trackers, their speed is limited by the model size. This creates clear motivation for developing compact models with similar performance as deep ones. Knowledge distillation [38], to be introduced in the next section, is a technique for achieving such goal.

1.5 Knowledge Distillation

Empirical evidence indicates that deeper neural networks have better performance on many problems and are easier to train alone compared to shallower networks. One typical example is the progression of the winners of the ImageNet [25] classification challenge towards deeper and deeper architectures over the past decade. However, it has been widely argued that neural networks may not have to be really deep. In [18], Cybenko et al. showed that a wide enough single-layer neural network with sigmoid activation has the capacity to approximate any decision threshold. Similarly, in [24], Dauphin et al. showed there are diminishing returns in training error when increasing the size of the neural network. While results the theoretical work implies the possibility of training light-weighted neural networks to achieve similar performance as a very deep one, what we have learned from practice is that such compact models are more difficult, or often infeasible, to train alone to get satisfying performance in the way we typically train neural networks, i.e. with an objective function specifying how to "penalise" the model for the discrepancy between its outputs and the groundtruth labels.

The term "knowledge distillation" is introduced by Hinton et al. [38]. The authors argued that the knowledge learned by deep neural networks is not the trained parameters of the network, but the mapping from the inputs to its outputs. For example, given two different images of cats, although a well-trained neural network may classify both as cats, it usually outputs different probability distributions over a set of classes for the two images. Compared to the groundtruth label, i.e. a one-hot encoded vector, such probability distribution has higher entropy (defined in Equation 1.4, where n is the number of possible values of a discrete random variable X, the p_X the probability distribution over X), thus carries more information (in information theory, entropy is a measure of the amount of information [84]).

$$\text{entropy} = \sum_{i=1}^{n} p_X(x_i) \log \frac{1}{p_X(x_i)} \tag{1.4}$$

Therefore, instead of using the groundtruth labels to train a small network, one can make use of the outputs of a well-trained deep network. Another intuition of the idea is that while the groundtruth labels assign the same probability (i.e. 0) to all the negative classes, the outputs of an accurate classifier do not. Instead, given an image of a cat, the probaility assigned to the tiger class is likely to be much higher than the probability assigned to birds, and the knowledge encoded by such difference is not available in the groundtruth labels but can be provided by a well-trained deep network. Such deep network used to provide the synthetic labels is called the “teacher” network, and the small network to be trained is called the “student” network. The output distributions by the teacher network are called “soft targets”, and are basically modified versions of the softmax output of the network, as shown in Equation 1.5,

$$p_i = \frac{\exp(z_i/T)}{\sum_j \exp(z_j/T)} \tag{1.5}$$

where p_i is the probability assigned to class i by the network, and z_i is the corresponding “logit”, which is the output of the linear operation by the corresponding neuron in the output layer of the teacher network, just before the softmax activation. The parameter T is called the “temperature” of the distribution, and when $T = 1$, Equation 1.5 becomes the standard softmax function.

The proposed distillation method in [38] matches the softmax outputs of the teacher (called “soft” targets) and the student raised to relatively high temperature, and at test time the temperature is reset to 1. Such high temperature T in the softmax function (Eqn.1.5) produces a softer distribution over the classes which help with the transfer of the knowledge from the teacher to the student. Such softer distribution means that the very small probabilities assigned to the negative classes (e.g. probability of 10^{-6} for the dog class predicted on a cat image) will have greater contribution to the cross-entropy loss, as illustrated in Figure 1.3, where it can be seen that at higher temperature the negative classes will have greater probabilities thus contribute more to the loss.

The combination of losses for matching the student output with the soft targets as well as the groundtruth targets, with the latter assigned smaller weight than the former, allowed the authors of [38] to train small networks that achieved higher classification accuracies than the teacher. In this project we devise similar knowledge distillation method to distil the state-of-the-art DCF tracker, ATOM. Although the domain is different, the idea of forcing the student

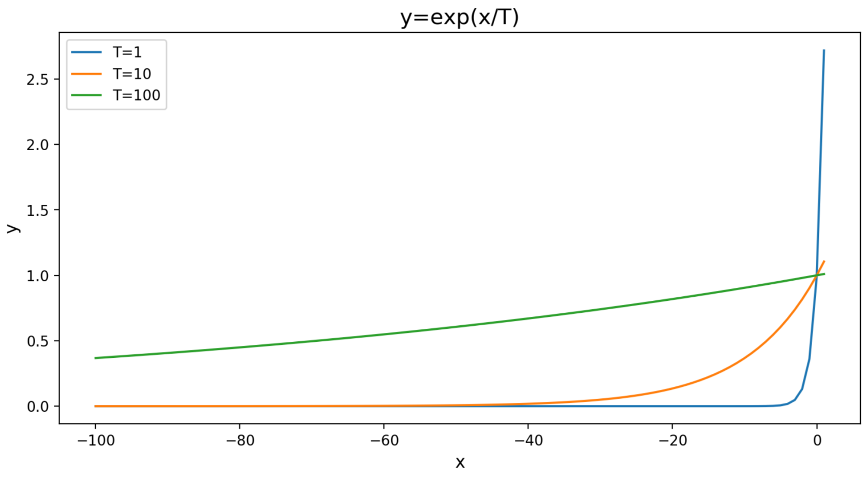

Figure 1.3: Illustration of exponential function raised to higher temperatures. Looking at x values from -100 to -80, it can be found that the magnitude and variation in y values become greater at higher temperatures.

to mimic the teacher can be inherited to create appropriate loss functions for distilling trackers. The exact details of our proposed method will be presented in Chapter 3, and before diving into the details, we review the existing literature on visual tracking and knowledge distillation.

1.6 Structure

The rest of this report is structured as follows. In Chapter 2, we review the existing literature on the related topics, i.e. visual object tracking and knowledge distillation. In Chapter 3 we firstly revisit the ATOM tracker to provide rationale behind our proposed method, and then formally introduce our method for distilling the ATOM tracker. We present our experiments in Chapter 4, including the methodology and the results obtained. Finally, we conclude our work in Chapter 5, giving a critical analysis of the advantages and drawbacks of our method, and discuss possible future work in this field.

Chapter 2

Related Work

In this chapter we review the related historic and more recent work in the visual tracking and knowledge distillation literature. Specifically, we first revisit some of the typical traditional tracking algorithms in Section 2.1, then in Section 2.2 and 2.3, we introduce some of the remarkable work in deep learning literature and take a look at the development of tracking algorithms under the influence of deep neural networks. Finally, we revisit the existing research on knowledge distillation and their applications in visual tracking.

2.1 Traditional Visual Tracking

Traditional tracking algorithms can be divided into generative methods and discriminative methods. Generative methods try to learn a generative model to encode the appearance information of the target and adaptively update the model as the appearance of the target varies. These generative models can be categorised into template-based, subspace-based, and sparse representation [58].

Template-based. These methods aim to develop template representations of the target using the visual information to match with the search image. In the very early work, Lucas et al. [64] proposed a solution to the image registration problem[1] based on spatial intensity gradient of the images. Following [64], templates based on raw intensity values have been widely applied. For example Hager et al. [32] developed an efficient method for calculating a "motion template" based on pixel brightness values that can be used to tackle appearance changes of the target caused by relative motion. To further improve the robustness of template-based methods to appearance changes, Matthews et al. [67] developed a method for updating the target template based on a correction mechanism that can be applied to keep the template up-to-date with appearance changes. Other than raw-pixel-based templates, other template representations of

[1]formulated as the problem of finding the location of the region in image A that is the most similar to an object B in [64]

targets have also been applied. The authors of [17] proposed a kernel-based method for matching colour-histogram templates. In [41], the authors proposed a model that measures "stability" of image structure during tracking and keeps an appearance model composed of stable image structures, and an online EM (Expectation-Maximisation) algorithm is used to update the appearance model.

Subspace-based. In order to further tackle the appearance changes during tracking, various subspace-based methods [79][59][97][96] have been developed. In [79] Ross et al. proposed the incremental visual tracking (IVT) algorithm, which makes use of incremental methods for principal component analysis (PCA) to online update a low-dimensional subspace representation of the target. Such subspace representation was shown to be effective in adapting to varying target appearance. Similarly in [96], Wang et al. characterised the target object using a 2D PCA-based subspace representation. While these methods use vector-based subspace representation, the authors of [59] argued that vector-based approaches introduce high spatial redundancies and proposed a tensor eigenspace representation that could be incrementally updated online. In [97], Wen et al. developed the weighted tensor subspace (WTS) representation of the target based on the raw pixel values and the Retinex algorithm [29][45], which is designed to produce illumination-insensitive features of given images.

Sparse representation. Trackers that use sparse representation are also called $\ell 1$ trackers, as they track the target by solving an $\ell 1$ norm related minimisation problem which typically resorts to "representing the target with a sparse approximation over a template set" [68][91][69][6][42][101]. Such tracker was firstly proposed by Mei et al. [68], who formulated tracking as a sparse approximation problem in which the target is represented sparsely by a set of trivial templates. The sparsity is achieved by minimising $\ell 1$-regularised projection error. Although the algorithm proposed was novel, it suffered from computational cost due to the number of $\ell 1$ objectives to minimise. Later in [69], Wei et al. improved the computational efficiency of their method by calculating a minimum error bound before sampling candidates for $\ell 1$ minimisation. To further improve the efficiency of solving the $\ell 1$ minimisation problem, Bao et al. [6] introduced the "accelerated proximal gradient approach" which resulted in real-time $\ell 1$ tracking. To tackle the problem of similar objects and occlusion, Jia et al. [42] proposed a tracking method that builds sparse appearance templates exploiting the local structure of the target, and their method combined the IVT algorithm and sparse representation for template updating.

Unlike the generative methods, discriminative methods pose tracking as a classification problem to distinguish the target from the background. Various classic machine learning algorithms have been applied to such classification, such as Support Vector Machine (SVM) [3], boosting [31][4], random forest [81]. In [3], Avidan integrated an SVM classifier into an optic-flow based tracker and replaced the original tracking mechanism (searching for image transformations that minimise the intensity difference between two frames) with SVM classification. The online boosting approach presented in [31] used AdaBoost algorithm to perform online selection and classification of the most discriminative features. AdaBoost was also adopted in [4] to combine weak classifiers that output confidence map on a given search frame into a strong one.

Since the introduction of Minimum Output Sum of Squared Error (MOSSE) formulation of discriminative correlation filter (DCF) based tracking by Bolme et al. [11], DCF tracking has gained increasing attention. Under this framework, a correlation filter can be efficiently trained in the Fourier domain using Equations 1.1 and 1.2. In [37], Henriques et al. proposed a kernelised correlation filter (KCF) based on circulant data matrix of translated images that showed similar computational efficiency as the basic linear correlation filters. Li et al. [61] presented the Scale Adaptive with Multiple Features (SAMF) algorithm to enable KCF tracker to deal with varying target scale. To better cope with scale variations, Danelljan et al. [21] trained separate filters for scale and translation estimation. Later in [22] it was argued by Danelljan et al. that the way DCF trackers were trained introduced undesirable boundary effects to the filters, so they proposed the Spatially Regularised Discriminative Correlation Filters (SRDCF) to alleviate this phenomenon.

The conventional tracking algorithms described above demonstrated great performance in their times, but most of them relied on hand-crafted features which potentially limited their performance. The revolutionary development of convolutional neural networks (CNNs) greatly enhanced the performance of both generative and discriminative methods. In the next section, we will highlight some remarkable development in these deep learning models so as to prepare for introduction of past work on deep visual tracking.

2.2 CNN Architectures

CNNs have revolutionised the area of computer vision with its strong ability to automatically learn rich feature representations of images. In essence, a basic CNN is simply layers of cross-correlation and non-linearity stacked one after another to map an input tensor (e.g. an image) to desired outputs. It can be seen as the outcome of encoding human knowledge as inductive bias to the hypothesis space, with the main focus to be on the translational invariance property, which means that the same response is produced for the same object regardless of its position in the image. The first piece of work that introduced CNNs under modern deep learning framework was presented by LeCun et al. in [53]. In their work, LeCun et al. developed the famous CNN architecture named LeNet-5 (shown in Figure 2.1) and firstly applied "back-propagation" [80] to train a CNN. Back-propagation refers to the use of the chain rule in calculus to efficiently calculate the derivatives of network parameters so as to optimise them using gradient-based methods such as gradient descent. The later development of CNNs have

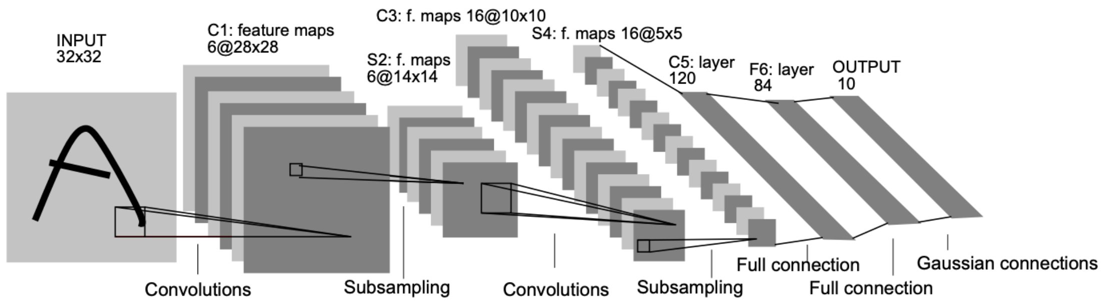

Figure 2.1: Architecture of Lenet-5 [53]. Image is taken from the original paper.

largely been stimulated by the ImageNet [25] competition where the task is to classify images among 1000 categories. One of the breakthroughs in early years of ImageNet is the introduction of AlexNet by Krizhevsky et al. [52] who trained the deep CNN with efficient use of GPUs. Later, Simonyan et al. [85] proposed the VGGnet that used very small convolution filter sizes and had 16 layers. The trend of going deeper continued with Szegedy et al. [88] introducing the GoogLeNet which had 22 layers. The main innovation in this network was the introduction of the "inception block" which concatenates convolution and pooling operations at different scale in one layer. To address the problem of vanishing gradient that makes deeper networks hard to train, He et al. [36] proposed the ResNet architecture composed of "residual blocks" that involve skip-connection from earlier layers directly to deeper layers so as to ease gradient flow in back-propagation. Such innovation allowed extremely deep networks to be feasible, e.g. the ResNet-101 that features 101 layers.

Although these deep architectures have been applied to many vision tasks other than image classification and have shown great performance [75][78][30][35], they are generally computationally heavy and infeasible to be implemented on CPU-only platforms. The MobileNets [39][83] are designed to have both strong representational power and low computational cost which is achieved by splitting the standard 2D convolution operation into depthwise convolution and 1×1 point-wise convolution. Such modification greatly reduces the computational cost while retaining the nature of CNNs. More details regarding MobileNets will be presented in Chapter 3.

The development of CNNs for the image classification problem has shown that these networks have strong representation power and their convolution filters learn to detect features that are general to many computer vision problems. Therefore, lots of work has been proposed in visual tracking to involve the advanced architectures built for image classification as part of the trackers. In the next section, we review such deep trackers.

2.3 Deep Visual Tracking

Similar to traditional tracking methods, deep trackers can also be divided into generative and discriminative models. While the former tends to deploy Siamese architecture to perform certain form of matching to track the target, the latter is mostly based on DCF.

2.3.1 Siamese Trackers

In [89], Tao et al. devised a two-stream CNN with fully-connected layers to learn matching functions between the target and search patches. Although satisfying performance was achieved, the tracker was designed to evaluate multiple search proposals in each frame so only ran at 2 FPS (frames per second). Later, Bertinetto et al. [8] introduced the SiamFC tracker that used CNNs to extract features and a correlation operation to perform searching. SiamFC takes as input the whole search image instead of proposed search patches, so in convolutional manner, it could evaluate all regions in the search image in one pass. As a result, SiamFC achieved much higher FPS of above 80. Following [8], various improved and modified versions of such Siamese network has been proposed [13][90][105][34][26][57][55].

Valmadre et al. [90] introduced a correlation filter in the target path of the Siamese network to learn a more discriminative template of the target, and the resulting CFNet was trained end-to-end treating the correlation filter as a differentiable layer. In order to make use of both semantic and appearance embedding of the target, He et al. [34] proposed a two-fold Siamese network that had one branch for matching the semantic features of images and another branch for matching appearance features of the objects. Although the two-fold tracker exceeded SiamFC in tracking accuracy, it suffered from the additional computational costs. To enhance the distinguishing power of the CNN backbone of Siamese trackers, Dong et al. [26] proposed a triplet loss for training Siamese networks.

More recently, Li et al. [57] proposed the SiamRPN tracker that combined SiamFC with the Region Proposal Network (RPN) introduced in [76] for object detection. Concretely, SiamRPN inherits the feature extraction Siamese network of SiamFC to obtain feature maps $\varphi(x)$ and $\varphi(z)$ for the target x and search image z. While these two feature maps are cross-correlated to obtain final confidence scores of object location in SiamFC, they are fed into the RPN in

SiamRPN which outputs a $w \times h \times 2k$ tensor ($A^{\text{cls}}_{w \times h \times 2k}$) denoting the probabilities of being the centre of the target and the opposite case for each of the k anchor boxes at each location, and a $w \times h \times 4k$ tensor ($A^{\text{reg}}_{w \times h \times 4k}$) that labels the position of the regressed bounding boxes for each anchor at each location, i.e.

$$A^{\text{cls}}_{w \times h \times 2k} = f_{\text{cls}}(\varphi(x)) \star f_{\text{cls}}(\varphi(z)) \tag{2.1}$$

$$A^{\text{reg}}_{w \times h \times 4k} = f_{\text{reg}}(\varphi(x)) \star f_{\text{reg}}(\varphi(z)) \tag{2.2}$$

where f_{cls} is the convolution filter used for classification, f_{reg} is the convolution filter used for bounding box regression, and $\star$ denotes cross-correlation operation. The introduction of RPN allowed SiamRPN to accurately regress the bounding box for the target, and made it the winner of VOT-2018 Real-time challenge. Later, Li et al. [55] presented an improved version of SiamRPN called SiamRPN++, which involved a very deep feature extraction backbone, ResNet-50, and performed well in VOT-2019 ranking top-10 in the real-time challenge. Most of the top-10 Siamese trackers in this challenge were based on SiamRPN++.

2.3.2 DCF Trackers

Most of the DCF-based deep trackers are developed by replacing the hand-crafted features used in traditional DCF tracking algorithms with CNN features. For example, in [23], the HoG (histogram of Gaussian) features used in SRDCF [22] was replaced by deep features extracted from early layers of a CNN, and these deep features showed better performance. Similarly in [66], CNN features at different scales (coming from different layers) are fed into a DCF for tracking, and the resulting tracker was called HCFT (Hierarchical Convolutional Features for Tracking). Qi et al. [74] also made use of features from different layers of a CNN, but different from HCFT[66], they trained many “weak” trackers which are based on different CNN layers and combined them using a Hedge algorithm they developed for tracking. The combined tracker showed superiour performance compared to HCFT. Danelljan et al. [20] proposed ECO (Efficient Convolution Operator) which is a factorised version of convolution to improve the computational efficiency of CNN-based trackers.

Other than feature extraction, deep CNNs have also been used as the discriminative classifier [92][72][71][15][33]. In [92], Wang et al. designed a CNN comprising part of the VGG-16 network pretrained on the ImageNet classification dataset for general feature extraction. This was then followed by a “general network” that builds general categorical representation of the search image and a “specific network” that extracts target-specific information. The target is then localised using the output feature maps of these two networks with a distractor detection mechanism. Similar idea of training domain-specific CNN classifiers was presented in the MD-

Net tracker [72]. MDNet (multi-domain network) consists of a pre-trained CNN for detecting general features and domain-specific output layers that are trained online to capture target-specific knowledge. The architecture of MDNet is shown in Figure 1.2. In [71], Nam et al. built a tree of multiple CNNs to maintain different appearance features and estimate target state in an ensemble manner. The resulting tracker TCNN won the VOT-2016 Short-term challenge. Han et al. [33] proposed a regularisation method named "Branchout" for emsemble tracking. In this method, a CNN-based tracker followed by multiple output branches was regularised by selecting only a subset of branches for online updating to keep up with the target appearance change.

More recently, a novel deep DCF tracking framework, ATOM (accurate tracking via overlap maximisation) inspired by the IoUNet [44] was proposed in [19]. In contrast to earlier DCF trackers in which bounding box regression was performed via a fully-connected regression head in a similar manner to the RPN [76], ATOM estimates the bounding box by maximising the overlap (Intersection-over-Union, or IoU score) between the proposed region and the target. This is achieved by offline pre-training an IoUNet that takes as input feature embeddings of the target and search images, the groundtruth bounding box A for the target image and a proposed bounding box B in the search image, and predicts the IoU score between the proposed box B and the groundtruth bounding box G in the search image. The groundtruth IoU score is defined in Eqn.2.3.

$$\mathrm{IoU}(B, G) = \frac{B \cap G}{B \cup G} \tag{2.3}$$

The feature extraction backbone used is ResNet-18. During online tracking, the parameters of the IoUNet and feature extractor are kept unchanged and the proposed box in the search image is regressed to maximise the IoU score. Other than the innovation in bounding box regression, ATOM also implemented Gauss-Newton method to efficiently train its DCF (a two-layer CNN) online.

Several modified versions of ATOM tracker appeared in the VOT-2019 challange [50]. DRNet, the winner of VOT-2019 Short-term challenge, introduced a distractor-aware loss in the online training of the DCF network. Compared to ATOM, DRNet used a much deeper backbone, ResNet-50, to achieve better tracking accuracies. Trackyou is the second best tracking in the short-term challenge, and modifies ATOM by offline pre-training the DCF classifier using triplet loss and binary cross-entropy loss. The third best tracker, ATP, improves tracking performance by combining ATOM with SiamMask [95] to produce segmentation of the estimated target region. Although achieving high accuracy and robustness when there was no speed requirement, these top-3 trackers in the short-term challenge ranked behind the Siamese trackers in the real-time challenge, mainly due to their heavy backbones. In the next section, the work done on knowledge distillation, a technique for compressing such heavy networks, is reviewed.

2.4 Knowledge Distillation

As introduced in Section 1.5, knowledge distillation was formulated by Hinton et al. [38] for distilling classifier networks by matching the softmax distributions output by the teacher and the student at high temperatures. Prior to [38], the pioneering work of Caruana et al. [5] showed that it is possible to transfer the knowledge learned by a large ensemble of networks to a single compact network. In their later work, Caruana et al. [5] proposed a compression method to compress a single large neural network, in which the small student network is trained to mimic the "logits" (the outputs of the final hidden layer of a classifier before softmax activation) produced by the teacher. Matching the logits instead of the softmax outputs has the effect of magnifying the contribution of the negative classes to the loss function, although it is shown in [38] that matching logits is a special case of matching softmax distribution at high temperature.

To compress CNNs, Romero et al. [77] developed the FitNets which are as deep as the teacher networks but have less filters at each layer. They trained the student at layer-level for mimicking the representations produced by the teacher at each layer. This was a rather strong restriction and the student network simply might not have the capacity to learn such knowledge. While in these works the student architectures have been selected manually, Chen et al. [14] developed the N2N algorithm to transfer the knowledge of a teacher to students with incrementally enlarged architectures. In such manner, the optimal architecture with the best trade-off between the gain in performance and loss in speed was selected.

Knowledge distillation has recently been applied to visual tracking [103][94][63]. In [103], Zhu et al. distilled a DCF tracker by matching the layer-wise responses of the student and the teacher backbone networks. They manually designed a 5-layer CNN architecture to distil the knowledge from a VGG-19 network pre-trained on image classification with ImageNet data. In specific, the last two layers of their student network had the dimensions same as the last two layers of VGG-19, i.e. the same height, width and depth, so their loss function directly measures the L2 difference between the last two layers of the student and the teacher.

Wang et al. [94] argued that the method presented in [103] did not bridge the gap between the pre-training task of object classification and the target task of visual tracking. Instead, they argued that the reduced architecture of the student may not have the capacity to mimic exactly the rich representations of the teacher that are built for object classification, so a better way to train the student is to also inject tracking-specific knowledge such that the objective of the student is adapted to tracking problem. Therefore, in addition to learning the feature representations produced by the teacher directly, they also devised a "tracking loss" defined as

$$\mathcal{L}_{\text{tracking}} = \|\mathbf{r} - \mathbf{g}\|^2 \tag{2.4}$$

$$\text{s.t.} \qquad \mathbf{r} = \mathcal{F}^{-1}(\hat{\mathbf{w}}^* \odot \hat{\varphi}(\mathbf{z})) \tag{2.5}$$

$$\hat{\mathbf{w}} = \frac{\hat{\mathbf{y}}^* \odot \hat{\varphi}(\mathbf{x})}{\hat{\varphi}(\mathbf{x}) \odot \hat{\varphi}^*(\mathbf{x})} \tag{2.6}$$

where $\mathbf{x}$ is the target image patch centred at the target object, and $\mathbf{z}$ is a search image patch with the same dimensions as $\mathbf{x}$. Eqn.2.6 is identical[2] to Eqn.1.2, which is used for solving the correlation filter with the groundtruth Gaussian label $\mathbf{y}$ and student feature embeddings (φ) of the target patch $\mathbf{x}$. The correlation filter $\mathbf{w}$ is then used to infer the confidence map $\mathbf{r}$ for the search patch $\mathbf{z}$ using Eqn.2.5. The tracking loss is then the L2 difference between the inferred map $\mathbf{r}$ and the groundtruth $\mathbf{g}$ for the search patch. With the tracking loss, the authors argue that one can modify the objective of the backbone from classification to visual tracking. However, it has not been studied if this method can be applied to ATOM-based trackers that may require more general feature representations. In addition, the student architecture was obtained by randomly pruning $7/8$th of filters in each layer, which might not be the optimised model.

The most recent work on distilling trackers is [63], where Liu et al. proposed a method to distil Siamese trackers. In addition to matching the bounding box coordinates and target probabilities output by the classification and regression heads of the student and the teacher, they also introduced a "Siamese Target Response" (STR) loss to match the layers of the backbones. In specific, they first obtain the features $\varphi_s(\mathbf{x}), \varphi_s(\mathbf{z})$ extracted by the student backbone on a target patch $\mathbf{x}$ (different from the previous patches mentioned, now $\mathbf{x}$ is a cropped patch to only include the target) and a search patch $\mathbf{z}$. The teacher also extract features $\varphi_t(\mathbf{x})$ and $\varphi_t(\mathbf{z})$. A function $F(U)$ is defined to obtain the channel-wise sum of absolute values of all channels of a feature map U. The features for the target are matched directly:

$$\mathcal{L}_{\text{x}}^{\text{STR}} = \|F(\varphi_s(\mathbf{x})) - F(\varphi_t(\mathbf{x}))\|^2 \tag{2.7}$$

and the features for the search patch are matched via

$$\mathcal{L}_{\text{z}}^{\text{STR}} = \|F(W_s \odot \varphi_s(\mathbf{z})) - F(W_t \odot \varphi_t(\mathbf{x}))\|^2 \tag{2.8}$$

$$\text{s.t.} \qquad W_s = \varphi_s(\mathbf{z}) \star \varphi_s(\mathbf{x}) \tag{2.9}$$

$$W_t = \varphi_t(\mathbf{z}) \star \varphi_t(\mathbf{x}) \tag{2.10}$$

[2]the operators have the same meaning as decribed for Equation 1.2

where the intended effect of the weight maps W_s, W_t is to let the student focus more on learning the feature for the target region in the search patch. The authors argued that this could be helpful in the presence of distractors. This method, although developed for Siamese trackers, can also be applied to distil the backbone network of DCF trackers. Also, the student architecture was obtained by incrementally trying different pruning options using deep reinforcement learning algorithms, but is still based on the network structure of the teacher.

In our work, we focus on distilling ATOM-based DCF trackers and propose a novel loss function to ease the training of the student backbone. Unlike the previous work where the student architecture is basically a reduced version of the teacher architecture, we deploy state-of-the-art compact CNN, MobileNetV2 [83] as the backbone of our student network. Our proposed distillation method, inspired by [94], jointly transfers the knowledge from the teacher to the student and compresses the student network. We also implement the existing methods and compare their performance with ours. In the next chapter, we introduce our proposed method formally.

Chapter 3

Proposed Method

3.1 Revisiting ATOM Tracker

In this section we provide a brief introduction of the ATOM tracker [19]. We refer the reader to [19] for more details of ATOM, but for convenience, we introduce the essential concepts before presenting our method. The overall architecture is shown in Figure 3.1. The tracker uses ResNet-18 pre-trained on ImageNet dataset as the feature extraction backbone, and has two components for tracking: an IoUNet for target state-estimation (where state refers to the bounding box), and a classifier to predict a coarse location of the target.

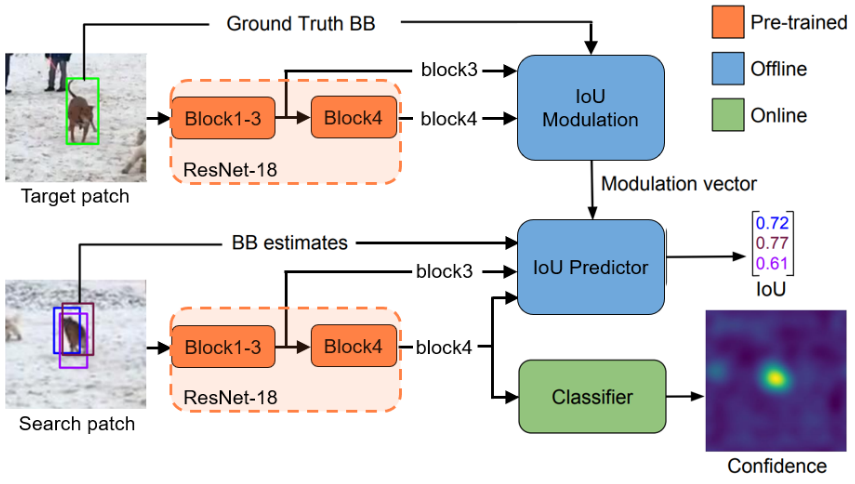

Figure 3.1: Overview of the ATOM tracker [19]. Image is taken from the original paper and modified.

IoUNet. The IoUNet is responsible for regressing the bounding box in a search image. It is designed to predict the IoU score (defined in Eqn.2.3) between a proposed bounding box B in a search image $\mathbf{z}$ and the groundtruth target bounding box G in $\mathbf{z}$. It consists of several convolutional and pooling layers, followed by fully connected layers for predicting the IoU score. Concretely, it can be broken down into two parts. The first part takes as input the multi-scale features ("block3" and "block4" of ResNet-18) extracted for a target image centred at the target as well as the groundtruth bounding box A in the target image and outputs a modulation vector that can be thought of as a 1D encoding of the target. The second part receives the features (again, “block3” and “block4” of ResNet-18) for the search image and a proposed bounding box B, then along with the modulation vector, it predicts the IoU between B and the target in the search image. The IoUNet is trained entirely offline, and during tracking, its parameters are frozen and the proposed box in the search image is optimised with gradient ascent to maximise the IoU output. We point the user to [19] for the exact architecture of the IoUNet.

Classifier. The goal of the classifier is to predict a rough location of the target in current search image. It is a CNN formulation of a DCF, and consists of two fully convolutional layers with parameters θ representing a function $f_\theta(\varphi(\mathbf{z}))$ that takes the feature embeddings $\varphi(\mathbf{z})$ (“block4” of ResNet-18 only) of a search image $\mathbf{z}$ as input and outputs a 2D confidence map for target location in the search image, and it is trained to minimise the objective

$$L(\theta) = \|f_\theta(\varphi(\mathbf{z})) - \mathbf{g}\|^2 + \lambda \sum_{\theta} \|\theta\|^2 \tag{3.1}$$

i.e. Eqn.1.3 plus regularisation term, where $\mathbf{g}$ denotes the groundtruth 2D Gaussian label and λ is the regularisation parameter. The classifier is trained entirely online with more efficient Gauss-Newton and conjugate gradient optimisation algorithms.

Online tracking. The overall tracking pipeline of ATOM is as follows. Given a target patch (the first frame) and the groundtruth target box, firstly the modulation vector $\mathbf{v}$ is extracted by the IoUNet. An initial training set $\mathcal{X}$ for the classifier is created by augmenting the target image and extracting the features using the ResNet-18 backbone. The corresponding 2D Gaussian labels $\mathcal{Y}$ are also created. The training set $\mathcal{X} \times \mathcal{Y}$ is used to train the classifier using Gauss-Newton and conjugate gradient method. Once the classifier is initialised, for each subsequent frame (search patch) $\mathbf{z}$, an image patch centred at previous target position with size corresponding to approximately 5^2 times the target area is sampled and feature $\varphi(\mathbf{z})$ is extracted for that patch. The feature is then fed into the classifier to produce 2D confidence map, from which a target position is estimated by finding the location in the image patch that corresponds to the peak of the map. With this estimated position and previous target size, a bounding box proposal is created, which is then used to generate 9 other boxes by adding Gaussian noise to the position and size of the box. The 10 proposed boxes, together with the feature map $\varphi(\mathbf{z})$ and the modulation vector $\mathbf{v}$, are fed into the IoUNet to be optimised for certain number of steps. Finally, the average of the top-3 boxes with highest predicted IoU scores is taken to be the inferred target box. During tracking, the classifier is updated using conjugate gradient method when a distractor peak is detected in the confidence map.

The two main sources of computation burden on the ATOM tracker are the ResNet-18 feature extraction backbone and the IoUNet. While the IoUNet is itself much smaller than the backbone, it is run forward and backward multiple times for bounding box optimisation as described above. The computations in the backbone are even heavier for some of the modified versions of ATOM (e.g. DRNet) that use the deeper ResNet-50. Therefore, our main focus is to devise loss functions to distil the well-trained backbone and IoUNet of the teacher tracker to smaller ones. We also reduce the size of the classifier accordingly to further reduce computational cost. However, the classifier is not distilled as it is trained online efficiently to learn discriminative features for specific targets. In order to verify that our method generalises to other ATOM-based trackers, we also evaluate our distillation method on DRNet [50], which is the winner of VOT-2019 Short-term challenge and shares the same architecture as ATOM as shown in Figure 3.1.

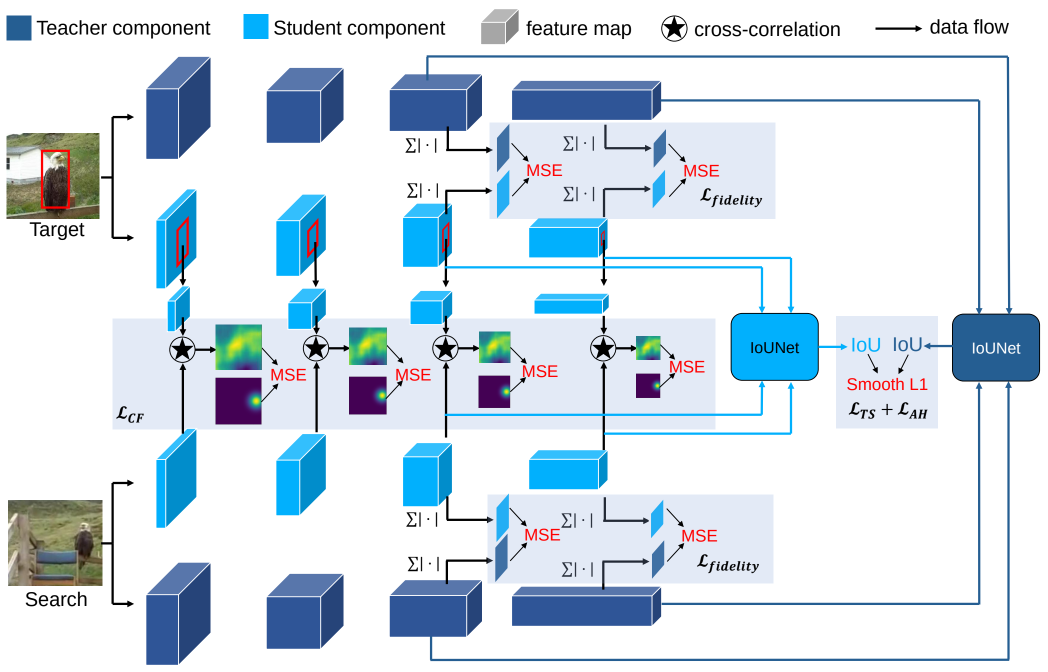

Figure 3.2: Pipeline of our proposed knowledge distillation method for ATOM. Each training instance consists of a pair of images. Fidelity and correlation filter losses are computed using the feature responses of the student and teacher; teacher soft loss and adaptive hard loss are computed to match their IoU outputs.

Our proposed method for distilling ATOM has four main components. Firstly, our student network deploys MobileNetV2 [83] that is developed specifically for embedded platforms as the feature extraction backbone to retain high representational capacity with very small size. The IoUNet and the classifier are also reduced in size in accordance with the smaller backbone. Secondly, we take the inspiration from [63] and devise losses for IoU predictions to distil the

IoUNet of the teacher network. Thirdly, we devise a fidelity loss and a correlation filter loss to transfer the knowledge from the ResNet-18 backbone of the teacher to the MobileNetV2 backbone of the student. Lastly, we combine the loss functions to end-to-end distil ATOM in a multi-task learning approach. The overall methodology is illustrated in Figure 3.2, where the cubic volumes in light and dark blue denote the feature maps produced by the student and the teacher backbones for the input images. From left to right, they are features from blocks 1 to 4 of the backbones, which will be specified in the following sections. In Sections 3.2 to 3.5, we introduce the four components in detail.

3.2 Student Architecture

In order to enable the student to run efficiently on CPUs while having strong capacity, we deploy the MobileNetV2 [83] as the feature extraction backbone of the student. While more details of MobileNetV2 can be found from the original paper, we introduce here briefly the way the network manages to reduce computations. The fundamental building block of MobileNetV2 is the bottleneck residual block, which consists of a depthwise separable convolution layer introduced in [39] and a linear bottleneck layer similar to [36]. In a full convolution operation, given an input feature map $\mathbf{F}$ of size $S_F \times S_F \times D_F$ where S_F defines the spatial resolution of $\mathbf{F}$ which is assumed to be square to ease explanation and D_F is the depth, each of the D_G kernels $\mathbf{K}$ of size $S_K \times S_K \times D_F$ slides along the spatial dimension of $\mathbf{F}$ and at each step the inner product is taken between the kernel and the overlapping part of $\mathbf{F}$ to produce a single value on the output feature map corresponding to the kernel. This is done for all D_G kernels to output a feature map $\mathbf{G}$ of size $S_G \times S_G \times D_G$. Therefore, the computational cost of a normal full convolution is proportional to

$$S_K^2 \times D_F \times S_F^2 \times D_G \tag{3.2}$$

A full convolution can be decoupled into two steps. Firstly, the input is filtered by the kernel in the sliding window fashion. Secondly, the filtered outputs are combined to produce a new feature map. In depthwise separable convolution, these two steps are separated. Firstly, a depthwise convolution layer performs channel-wise convolution to the input, i.e. D_F 2D kernels (each of size $S_K \times S_K$) are applied to the input with each of them convolved with each of the D_F input channels to produce an intermediate representation of size $S_G \times S_G \times D_F$. Then D_G 1×1 convolution (called point-wise convolution) is performed to combine the channels of the intermediate representation, giving rise to the output feature map $\mathbf{G}$ of size $S_G \times S_G \times D_G$. Because of such separation between the filtering and combination steps, the computational cost of depthwise separable convolution reduces to

$$S_K^2 \times D_F \times S_F^2 + D_F \times S_F^2 \times D_G \tag{3.3}$$

where the first term is for depthwise convolution and the second term is for point-wise convolution. It can be seen that the cost of the latter is much higher for high-dimensional output features (where dimension refers to depth, or number of channels). Now taking the ratio between Eqn.3.3 and 3.2 we find that the computation is reduced by a factor of

$$\frac{1}{D_G} + \frac{1}{S_K^2} \tag{3.4}$$

The depthwise convolution and point-wise convolution constitute the basic building block of MobileNetV2: the bottleneck residual block, shown in Figure 3.3. The operations in the bottleneck block is shown in Figure 3.4. It takes as input a low-dimensional (k) (dimension here refers to the depth of the feature map) feature map, and applies 1×1 convolution with ReLU activation to produce higher dimensional (tk, where t is called the expansion factor) representation. Next, 3×3 depthwise convolution with ReLU activation and stride s is performed for filtering, and the resulting feature map is projected to a lower-dimensional (d) subspace by a 1×1 convolution without non-linear activation. The idea of this design is the assumption that the feature maps can be embedded in some low-dimensional subspace where most information is preserved, but applying non-linear activation in low dimensions results in loss of information, which can be mitigated by first creating higher dimensional representation then applying the non-linearity. As such, the authors of MobileNetV2 argue that MobileNetV2 preserves higher expressiveness whilst reducing the amount of computation.

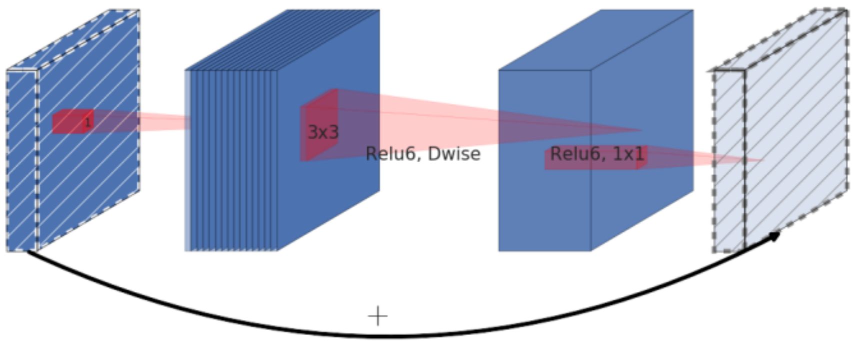

Figure 3.3: The bottleneck residual block, where a low-dimensional input is first convolved with 1×1 kernels (point-wise convolution) and applied ReLU activation to produce a high-dimensional feature map, which is then fed into depthwise separable convolution layer with ReLU activation. Finally, another 1×1 convolution with linear activation is performed to produce lower dimensional output. Image is taken from the original paper [83].

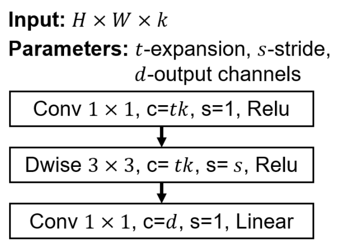
Input: $H \times W \times k$
Parameters: t-expansion, s-stride, d-output channels
Conv 1×1, c=tk, s=1, Relu
Dwise 3×3, c= tk, s= s, Relu
Conv 1×1, c=d, s=1, Linear

Figure 3.4: Operations in the bottleneck residual block, where c denotes output channel of each layer, s is the stride, and "Relu" and "Linear" are activation functions. Bottleneck block is characterised by the expansion factor t that determines the dimension of the intermediate representation, stride s and output dimension d.

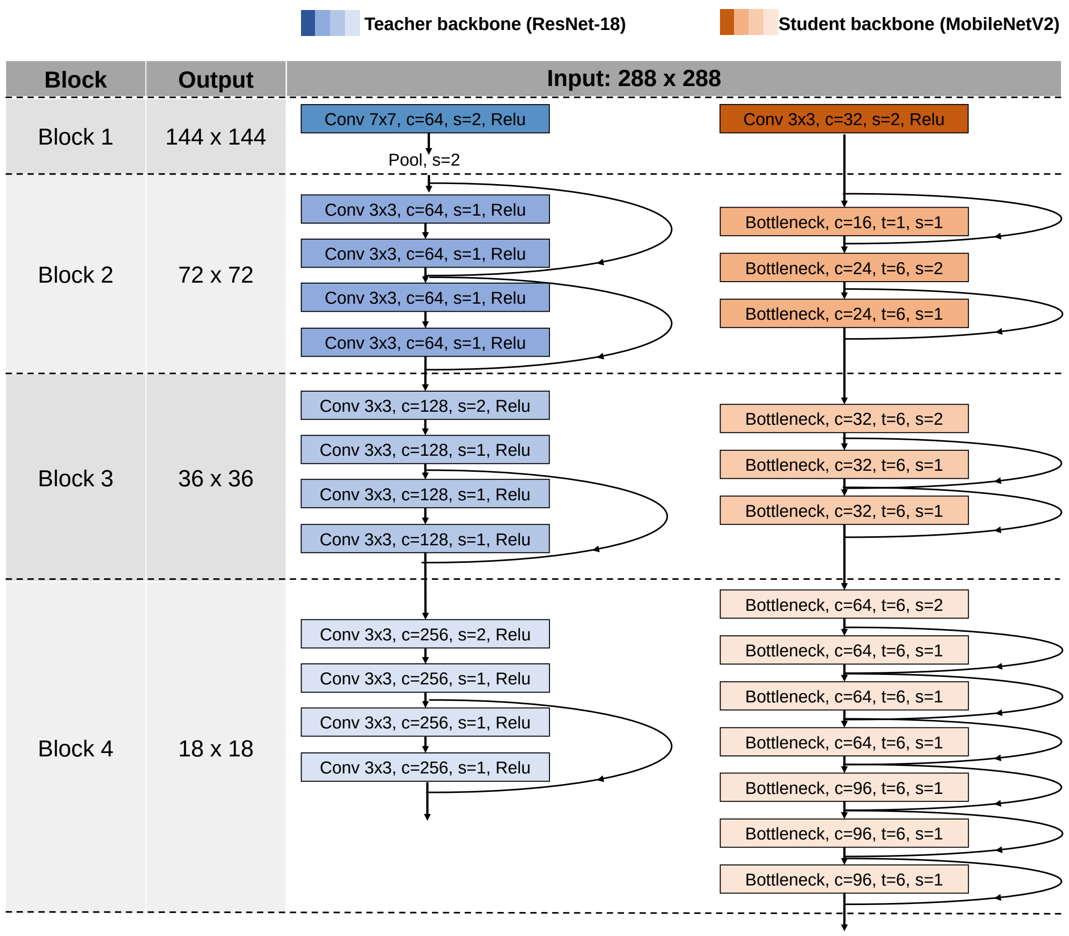
Teacher backbone (ResNet-18)
Student backbone (MobileNetV2)
Block
Output
Input: 288 x 288
Block 1
144 x 144
Conv 7x7, c=64, s=2, Relu
Pool, s=2
Conv 3x3, c=32, s=2, Relu
Block 2
72 x 72
Conv 3x3, c=64, s=1, Relu
Conv 3x3, c=64, s=1, Relu
Conv 3x3, c=64, s=1, Relu
Conv 3x3, c=64, s=1, Relu
Bottleneck, c=16, t=1, s=1
Bottleneck, c=24, t=6, s=2
Bottleneck, c=24, t=6, s=1
Block 3
36 x 36
Conv 3x3, c=128, s=2, Relu
Conv 3x3, c=128, s=1, Relu
Conv 3x3, c=128, s=1, Relu
Conv 3x3, c=128, s=1, Relu
Bottleneck, c=32, t=6, s=2
Bottleneck, c=32, t=6, s=1
Bottleneck, c=32, t=6, s=1
Block 4
18 x 18
Conv 3x3, c=256, s=2, Relu
Conv 3x3, c=256, s=1, Relu
Conv 3x3, c=256, s=1, Relu
Conv 3x3, c=256, s=1, Relu
Bottleneck, c=64, t=6, s=2
Bottleneck, c=64, t=6, s=1
Bottleneck, c=64, t=6, s=1
Bottleneck, c=64, t=6, s=1
Bottleneck, c=96, t=6, s=1
Bottleneck, c=96, t=6, s=1
Bottleneck, c=96, t=6, s=1

Figure 3.5: The student and teacher backbone architectures, where each "Bottleneck" block is specified in Fig.3.4. The blocks of MobileNetV2 are created by matching the spatial dimension of the output feature maps.

Empirical evidence in [83] shows that MobileNetV2 achieves state-of-the-art performance in many vision tasks while significantly reducing the computational costs compared to heavier architectures such as VGG-16, ResNet-50, etc. While the complete architecture involves 19 bottleneck residual blocks, we prune the network such that only the part whose output spatial dimension matches that of the teacher backbone is kept. The resulting architecture of the student as well as that of the teacher (ResNet-18) backbones are shown in Figure 3.5 for comparison.

For our purpose, we group the layers of MobileNetV2 into 4 blocks corresponding to the 4 blocks of ResNet-18 shown in Figure 3.1, such that the spatial dimensions (i.e. height and width) of the outputs of the corresponding teacher and student blocks are matched. In the rest of this report, we refer to the blocks of these networks instead of specific layers.

Besides the feature extraction backbone, the IoUNet and the classifier head of the student ATOM tracker are also different from those of the teacher. Specifically, the IoUNet of the student preserves the overall architecture of the teacher IoUNet, but the number of kernels in each convolutional layer and the number of neurons in each fully-connected layer are reduced by $1/4$. For the classifier, since it only has two convolutional layers, the number of kernels in each layer is reduced only by $1/2$ to retain reasonable capacity. While the backbone and the IoUNet are distilled offline, the classifier is trained online using the original method introduced in ATOM. As a result of such architecture reduction, our student ATOM tracker is only 18MB compared to 109MB of the teacher (83%) decrease. In the following sections, we first present our method for distilling the backbone in Section 3.3, then introduce the loss functions for distilling the IoUNet in Section 3.4.

3.3 Distillation of Backbone

The role of the backbone is to extract features from the given frame that can be used by the IoUNet and the classifier to track the target, so the student backbone should be able to produce semantic representations as close as possible to the teacher backbone. For this purpose we use a fidelity loss ($\mathcal{L}_{\text{fidelity}}$) to match the feature responses of the student to the teacher. However, due to the reduced capacity, the student backbone may not be able to perfectly mimic those representations, so we devise a correlation filter loss ($\mathcal{L}_{\text{CF}}$) that emphasises locating of the target to inject tracking-specific knowledge to the student and aid the learning process. The details of these loss functions are presented in what follows.

3.3.1 Fidelity Loss

To transfer the knowledge of the teacher backbone to the student backbone, we use the fidelity loss to match their feature responses. As shown in Figure 3.2, the fidelity loss is computed using the features extracted from the last two blocks (i.e. block 3 and block 4) of the backbones. Specifically, given a target patch $\mathbf{x}$ and a search patch $\mathbf{z}$, the student backbone extracts features $\varphi_i(\mathbf{x})$ and $\varphi_i(\mathbf{z})$, and the teacher backbone extracts $\psi_i(\mathbf{x})$ and $\psi_i(\mathbf{z})$, where $i \in \{1, 2, 3, 4\}$ indexes the block number. Although the spatial dimensions of the features of the student and the teacher at the same block match, their depth do not. We use the method introduced in [63] and define

$$F(U) = \sum_{c=1}^{C} |U_c| \tag{3.5}$$

where U is a feature map of dimension $H \times W \times C$, and $U_c \in \mathbb{R}^{H \times W}$ is a single channel of U. Then for each input image, we stack the features of the same blocks of the student and the teacher with F to produce 2D features, then L2 norm of the difference is taken to compute the fidelity loss. Formally,

$$\mathcal{L}_{\text{fidelity}} = \sum_{i=3}^{4} \|F(\varphi_i(\mathbf{x})) - F(\psi_i(\mathbf{x}))\|^2 + \|F(\varphi_i(\mathbf{z})) - F(\psi_i(\mathbf{z}))\|^2 \tag{3.6}$$

The fidelity loss only matches the block 3 and block 4 features between the student and the teacher because they are deemed as the output layers in ATOM, and those features are used by the IoUNet and the classifier directly to infer target location. If the features at all blocks are matched, the restriction on the student network parameters would be too tight, and some of the layers may not have the capacity to perform the required mapping. Instead, matching only the output features with fidelity loss loosens the restrictions and allows the student backbone to learn internal representations adaptively to produce output features that are as similar as possible to the output features of the teacher.

3.3.2 Correlation Filter Loss

Considering the reduced capacity of the student network compared to the teacher, we propose the correlation filter loss (CF loss) in addition to the fidelity loss for training the student backbone. As shown in Figure 3.2, CF loss is computed using the features from all four blocks of the student extracted from the target patch $\mathbf{x}$ and search patch $\mathbf{z}$. For each block i, we first obtain the features $\varphi_i(\mathbf{x})$ and $\varphi_i(\mathbf{z})$. Then the features corresponding to the target in the target patch, $\varphi_i(\hat{\mathbf{x}})$, is obtained by applying precise ROI (region of interest) pooling [44] to $\varphi_i(\mathbf{x})$. The search patch features $\varphi_i(\mathbf{z})$ is then padded and cross-correlated with $\varphi_i(\hat{\mathbf{x}})$ to obtain a 2D output Q that has the same height and width as $\varphi_i(\mathbf{z})$. Due to the property of cross-correlation operation, the 2D map Q_i should have the highest value at the position that corresponds to the region in $\varphi_i(\mathbf{z})$ most similar to $\varphi_i(\hat{\mathbf{x}})$, i.e. it should peak at the location of the target in the search patch. Therefore, $\bar{Q}_i$ is obtained by dividing Q_i by its maximum value such that $\max \bar{Q}_i = 1$, and $\bar{Q}_i$ is matched using L2 loss with the groundtruth label G_i, which is a 2D Gaussian function centred at the target position in $\mathbf{z}$ with maximum value of 1. We match $\bar{Q}_i$ instead of Q_i with G_i because while G_i has maximum value of 1 (due to the way we generate it), Q_i does not, so matching Q_i directly with G_i will inevitably change the scale of the features learned, which is not desirable. Formally,

$$\begin{aligned} \mathcal{L}_{\text{CF}} &= \sum_{i=1}^{4} \|\bar{Q}_i - G_i\|^2 \\ \text{s.t.} \qquad \bar{Q}_i &= \frac{Q_i}{\max Q_i} \\ Q_i &= \varphi_i(\mathbf{z}) \star \varphi_i(\hat{\mathbf{x}}) \end{aligned} \tag{3.7}$$

where index i indicates the block of the backbone. The groundtruth labels G_i are centred at the same location correponding to target position in the search patch, but they has different sizes at different blocks, thus the indexing.

The CF loss has three effects. Firstly, the loss penalises the network for not producing target features that, when cross-correlated with the search patch features, do not produce a 2D map that indicates the target position in the search patch. Such penalty enforces the backbone to learn representations of the target that allow it to perform correlation-filter tracking. That is to say, it has to capture high level abstraction of the target appearance such that under appearance changes it can still produce robust and similar features for the target. Secondly, matching the output of cross-correlation between target and search patch features with the groundtruth Gaussian map that only peaks at target location forces the feature extractor to distinguish the target from the distractors in the search frame, for which the corresponding value in the correlation output can be high. Finally, it emphasises the learning of target features in earlier blocks of the student backbone such that if those blocks cannot mimic perfectly the corresponding features of the teacher, they can focus on learning robust representations for the target to reduce the CF loss.

3.4 Distillation of IoUNet

The IoUNet of ATOM has only one output, the IoU score predicted for the given bounding box in the search patch. We use two loss functions to match the IoU outputs of the student and the teacher. Firstly, we directly match the outputs in a regression fashion using the teacher soft loss (TS loss, $\mathcal{L}_{\text{TS}}$). Secondly, to make use of the groundtruth labels from the training data, we take the inspiration from [63] and devise an adaptive hard loss (AH loss, $\mathcal{L}_{\text{AH}}$). The details are provided below.

3.4.1 Teacher Soft Loss

The teacher soft (TS) loss follows the idea of the original distillation paper [38], where it is argued that the smaller student network is easier to train with the labels provided by the teacher network that have certain regularisation effect. Therefore, we use TS loss to match the IoU outputs directly:

$$\mathcal{L}_{\text{TS}} = L_1(\zeta_s, \zeta_t) \tag{3.8}$$

where ζ_s and ζ_t are the IoUNet outputs of the student and the teacher respectively, and L_1 is the smooth L1 loss defined as

$$L_1(x, y) = \begin{cases} \frac{1}{2}(x-y)^2 & \text{if } |x-y| < 1 \\ |x-y| - \frac{1}{2} & \text{otherwise} \end{cases} \tag{3.9}$$

3.4.2 Adaptive Hard Loss

Authors of [63] proposed to adaptively penalise the student using the groundtruth in addition to the teacher soft labels, so as to aid the learning process. We also find such method effective and devise the adaptive hard (AH) loss:

$$\mathcal{L}_{\text{AH}} = \begin{cases} L_1(\zeta_s, g) & \text{if } L_1(\zeta_t, g) - L_1(\zeta_s, g) < t \\ 0 & \text{otherwise} \end{cases} \tag{3.10}$$

where g is the groudtruth IoU label and t is a threshold set manually. The intuition is that when the student performs worse than the teacher, we use the groundtruth labels to aid the learning, and once the student outperforms the teacher by t, we stop the loss to avoid potential overfitting.

3.5 Multi-task Learning

We create a single loss function called Correlation Filter Knowledge Distillation loss (CFKD loss, $\mathcal{L}_{\text{CFKD}}$) that combines the four losses introduced previously (Equations 3.6, 3.7, 3.8, 3.10) to train the student model:

$$\mathcal{L}_{\text{CFKD}} = \omega_1 \mathcal{L}_{\text{fidelity}} + \omega_2 \mathcal{L}_{\text{CF}} + \omega_3 \mathcal{L}_{\text{TS}} + \omega_4 \mathcal{L}_{\text{AH}} \tag{3.11}$$

where $\omega_1, \omega_2, \omega_3, \omega_4$ are the weights associated with the individual losses. That is to say, the errors produced by the IoUNet outputs also flow into the backbone during back-propagation. Our empirical evidence show that such multi-task learning procedure improves the performance of the trained student model. The backbone and IoUNet of the student ATOM tracker is trained entirely offline with the CFKD loss using the off-the-shelf ATOM tracker provided in [19]. Once the student has been trained, during online tracking, the parameters of the backbone and IoUNet are frozen and the tracking algorithm is unchanged compared to the original ATOM tracker, with the classifier trained online using the efficient algorithm proposed in [19]. This concludes our proposed method, and we present our experiments in the next chapter.

Chapter 4

Experiments

We present the details of our experiments as well as the results in this chapter. In order to validate the effectiveness of different components of our method, we first present an ablation study of our CFKD distillation method on ATOM. Then we compare CFKD with existing competitive methods for distilling trackers including those proposed by Wang et al. [94] and Liu et al. [63]. To show that our method generalises to other trackers from ATOM family, we use it to distil DRNet, the winner of VOT-2019 challenge, and present the results. Finally, we evaluate our distilled models on popular tracking benchmarks and provide quantitative analysis of the advantages and drawbacks of our method.

4.1 Experiment Setup

4.1.1 Model Training

Training datasets. We use the training splits of the state-of-the-art corpus of video sequences for visual tracking, including TrackingNet [70], Large-scale Single Object Tracking (LaSOT) [28] and the COCO sequence dataset [62], to perform distillation for all of our experiments, which results in more than 37 million frames and forms the same training set used to train ATOM in [19]. Although with knowledge distillation the amount of data available to train the student is greatly increased because the groundtruth labels are no longer needed and the teacher acts as the labeller of the data, we still only used the same training set that is used to train the teacher. This is because the training set comprises two largest tracking datasets available and it is already significant amount of data for our student who has much smaller model size compared to the teacher, so we do not concern creating a larger training set.

Data preprocessing. Our method of sampling and pre-processing training instances follows that of ATOM, except that we do not perform data augmentation as it is less likely that our reduced model overfit to the large amount of data. Although details can also be found in [19], we re-iterate the process here for completeness. For each training instance, we first sample a pair of images from the video sequences separated less than 50 frames away from each other and set one of them as the target image and the other as the search image. Then from the target image, using the available groudtruth bounding box data, we crop a square patch centred at the target position that has area of approximately 25 times the target area (computed from the groundtruth bounding box) and resize the patch to 288×288 pixels to get a target patch $\mathbf{x}$. We find a square patch from the search image centred at the target in the same way but add Gaussian noises to the position and scale of that patch to obtain another square patch, which is cropped and resized to 288×288 to be used as the search patch $\mathbf{z}$. As such, we have a target patch centred at the target, and a search patch in which the target might have different appearance, scale and position. For the training of the IoUNet, in addition to the groundtruth bounding box in the target patch, we generate 16 candidate bounding boxes in the search patch by adding Gaussian perturbation to the groundtruth one and calculate their IoU scores with the groundtruth bounding box in the search patch. Therefore, each training instance comprises a pair of image patches, a groundtruth bounding box in the target patch, and 16 candidate bounding boxes in the search patch along with their IoU labels.

Optimisation and hardware. During training, we first load the teacher network and freeze all of its parameters, and the parameters of the student network are all trainable. The loss function used is the CFKD loss in Equation 3.11, with hyperparameters $\omega_1 = 100, \omega_2 = 0.1, \omega_3 = 1, \omega_4 = 0.1$. We use the batch size of 32 and train the student for 50 epochs, with 1000 batches sampled for each epoch. The optimiser used is ADAM [47], and the learning rate is decayed every 15 epochs exponentially from 10^{-2} to 10^{-5}. Our code is developed with PyTorch under the PyTracking framework provided by Danelljan et al.. The GPU used for training and evaluation is Nvidia GTX Titan X, on which the models are trained for around 25 hours. For the CPU evaluation, we run the trackers on a single-core 2.3GHz Intel Xeon CPU. It should be noted that the precise ROI pooling used by ATOM is not supported on CPUs, so when evaluating on CPUs, we replace precise ROI pooling with the standard ROI pooling, and no obvious effect on the performance is observed.

4.1.2 Model Evaluation

Benchmarks. Five benchmark datasets are used to evaluate our model, including OTB-100 [98], LaSOT [28], TrackingNet [70], VOT-2018 [48] and VOT-2019 [50], which are widely considered state-of-the-art in visual tracking. For our ablation study, we use OTB-100 as the validation set. This is because OTB-100 contains 100 trackings sequences that cover a wide range of challenging scenarios including illumination variation, occlusion, deformation, low-resolution, etc. The performance on OTB-100 is considered representative of general tracking performance. When comparing our method with existing distillation methods, we use TrackingNet and LaSOT in addition to OTB-100 to obtain more convincing results. Results of our distilled trackers on all of the five aforementioned test sets are also shown and compared to the state-of-the-art trackers on those datasets. OTB-100, TrackingNet, and LaSOT all use one-pass evaluation (OPE) strategy, in which the tracker is initialised and given the groundtruth bounding box in the first frame, then tracks all of the subsequent frames in the sequence continuously regardless of tracking outcomes. In contrast, the VOT challenges adopt reset-based method where the tracker is again initialised for the first frame but also re-initialised after 5 frames of a failure. The failure is detected when the predicted bounding box has zero overlap with the groundtruth.

Metrics. The OTB-100, TrackingNet and LaSOT all use success plots to evaluate tracking performance, with the area-under-curve (AUC) scores calculated from these plots used as the comparison metric. While OTB-100 also uses precision plots, TrackingNet and LaSOT adopt normalised precision as another metric. The VOT challenges use accuracy, robustness and expected average overlap (EAO) scores for evaluation. The details of these metrics are introduced below.

- **Success.** Also called OP_T (Overlap Precision at threshold T), success is defined as the proportion of successfully tracked frames in a sequence given a threshold for the IoU (intersection-over-union) score (in pixels), which is calculated between a proposed bounding box B_{pred} and the groundtruth box B_{gt} with

  $$\text{IoU} = \frac{B_{\text{pred}} \cap B_{\text{gt}}}{B_{\text{pred}} \cup B_{\text{gt}}} \tag{4.1}$$

  and a frame is considered tracked successfully if the IoU score of B_{pred} is greater than a pre-defined threshold T. The success plot is generated by computing the average success scores on all sequence at thresholds from 0 to 1, and the AUC score is defined as

  $$\text{AUC} = \int_0^1 \text{OP}_T dT \tag{4.2}$$

  In the rest of this report, when we refer to the AUC score without mentioning the success or precision plots, it is the AUC for the success plot.

- **Precision.** This metric measures the distance error (in pixels), which is defined as the Euclidean distance between the predicted target centre C_{pred} and the groundtruth C_{gt}:

  $$d = ||C_{\text{pred}} - C_{\text{gt}}||_2 \tag{4.3}$$

  Precision of a tracker on a sequence is defined as the proportion of frames where the distance error is lower than a certain threshold. The precision plot is then generated by computing the average precision on all sequences for a range of thresholds, and precision is reported in the AUC score of the precision plot. Since the precision plot only concerns the distance between the predicted and groundtruth bounding boxes, the success plot, who also considers the overlap between the boxes, is preferred over the precision plot [98].

- **Normalised precision.** Since the distance error (Eqn.4.3) is measured in pixels, precision is sensitive to the resolution of images and the target size. For example, while a certain distance error is regarded small for two large bounding boxes, it can mean significant degradation in performance for two small bounding boxes. Therefore, Muller et al. [70] proposed the normalised precision score where the distance error is calculated with

$$d_{\text{norm}} = \|D(C_{\text{pred}} - C_{\text{gt}})\|_2 \qquad \text{s.t.} \quad D = \begin{bmatrix} 1/W_{\text{gt}} & 0 \\ 0 & 1/H_{\text{gt}} \end{bmatrix} \tag{4.4}$$

  where W_{gt} and H_{gt} are the width and height of B_{gt}. And the normalised precision is computed in the same way as precision but with the modified distance error.

- **Accuracy.** Accuracy is obtained by computing the average IoU score between the predicted bounding boxes and the groundtruth boxes for successfully tracked frames (where the IoU between predicted bounding box and the groundtruth is greater than 0) in a sequence. And the final accuracy of a tracker is the mean of the accuracies for all sequences. The VOT-2018 and VOT-2019 challenges adopt reset-based evaluation strategy, in which the tracker is re-initialised 5 frames after a failure, where a failure is reported if the predicted bounding box has 0 IoU with the groundtruth. Therefore, 10 frames after such re-initialisation are not included when computing the accuracy.

- **Robustness.** Robustness is another basic measure that complements accuracy in VOT challenges. Since accuracy does not consider tracking failures, robustness reports the failure rate during tracking of a sequence.

- **Expected average overlap (EAO).** EAO, first presented in [49], is the main metric used by VOT-2018 and VOT-2019 to rank trackers. It is a measure that combines accuracy and robustness and indicates the average overlap of the predicted bounding boxes and the groundtruth bounding boxes a tracker is expected to attain on a large set of sequences. We point the readers to [49] for more details of how EAO is computed. We use the implementation provided with the VOT toolkit to compute EAO, accuracy and robustness.

4.2 Ablation Study

In this section we evaluate the effectiveness of different components of our method. As explained above, We use OTB-100 [91] as the validation set to compare different choices for the components.

4.2.1 Model Compression

We first examine the extent to which our method compresses the original ATOM tracker and the degradation in tracking performance brought by such compression. Table 4.1 compares the model complexity, tracking speed and performance on OTB-100 (in terms of AUC score defined in Equation 4.2 for one-pass evaluation) of the teacher ATOM tracker and the student ATOM tracker which we name mATOM short for “mobileATOM”. in addition to model size and speed, we also include another measure of model complexity, number of floating point operations (FLOPs). As the name indicates, it measures the number of operations between float numbers in a single run of a model, which in our case is tracking of one frame. The FLOPs in one convolutional layer is computed with Equation 4.5 (assuming the same height and width for the input):

$$\text{FLOPs} = S_K^2 \times D_F \times S_F^2 \times D_G + S_G^2 \times D_G \tag{4.5}$$

where S_K, S_F, S_G are the side lengths of the kernel K, the input feature map F, and the output feature map G respectively, D_F is the depth of the input, and D_G is the depth of the output. The first term in Equation 4.5 is the same as Equation 3.2, and the second term is for the bias operation.

	Backbone	Size	FLOPs	Backbone FPS	Tracking FPS	OTB-100 AUC (%)
ATOM	ResNet-18	109 MB	3.35 B	80 / 15	30 / 9	66.6
mATOM	MobileNetV2	18 MB	0.33 B	80 / 40	38 / 20	62.4

Table 4.1: Comparison of model size, complexity, speed and tracking performance of the teacher ATOM tracker and the student obtained with our CFKD method. Number of floating point operations (FLOPs) is reported in billions. The speed is reported in typical frames per second (FPS) on GPU/CPU. Backbone FPS means feature extraction FPS. AUC is defined in Eqn.4.2.

It can be seen from Table 4.1 that our distillation method compresses the model size of ATOM by around 6 times and reduces the amount of operations by more than 10 times. However, despite the reduction in the amount of computation, the student model does not have obvious improvement in speed compared to the teacher on GPUs, with the backbone MobileNetV2 extracting features at the same speed (80 FPS) as the ResNet-18 backbone of the teacher. This has also been encountered by Orsic et al. [73] and they conclude that while modern libraries for GPU (cuDNN) does not support the depthwise separable convolution which is the main contributor of the reduced computations in MobileNetV2. In other words, while modern GPU firmware provides optimisation for normal convolution operations, MobileNetV2 does not benefit from it, thus the little improvement in speed of the tracker on GPUs. Note that the sligtly faster tracking speed of the student on GPU is because the ATOM tracking algorithm involves multiple forward and backward passes through the IoUNet for bounding box regression, and the drastic reduction in computational costs of the student IoUNet (as described in Section 3.2) brings some improvement (8 more frames per second) in tracking speed on GPUs.

In contrast, on CPUs, where the parallel computational power of GPUs is absent, the reduced computation is clearly reflected by the enhanced speed. Now the student backbone extracts features at 40 FPS compared to 15 FPS of the teacher backbone. More importantly, while the teacher achieves only 9 FPS while tracking on CPU, the student achieves 20 FPS (more than 200% increase), which is considered real-time, with only 6% drop in tracking performance on OTB-100, i.e. our method results in a tracker that can achieve state-of-the-art performance while being able to track in real time on a single-core CPU. Although the competing Siamese trackers (e.g. SiamRPN [56], SiamFC [8]) that can achieve above 90 tracking FPS on GPUs also seem promising in real-time tracking on CPUs, study [40] has shown that they cannot. Therefore, with our CFKD distillation strategy, we trained a state-of-the-art DCF-based tracker that tracks faster than the Siamese trackers on CPUs.

4.2.2 Loss Functions

In order to study the impact of the four components of the CFKD loss in Equation 3.11, we distil ATOM with different combinations of these loss functions and evaluate the resulting models on OTB-100. The results are shown in Table 4.2.

We can see from Table 4.2 that when training the student network using only the groundtruth IoU labels, there is a significant drop in performance compared to the teacher, with about 25% decrease in both precision and success AUC scores. Using the teacher's soft IoU labels, i.e. the teacher soft loss, such degradation is alleviated, showing that the student is indeed easier to train with knowledge distillation method. When the groundtruth labels are utilised adaptively in addition to the soft labels (i.e. TS combined with AH), we again see slight increase in scores. However, when the fidelity loss is used together with TS and AH losses, the improvement in the student's performance is much more significant, indicating the importance of matching the feature representations of the student to those of the teacher. The performance is further boosted when CF loss is also included (i.e. with TS+AH+fidelity+CF), achieving

	GT	TS	AH	Fidelity	CF	multi-task	precision (%)	success (%)
Student	✔					-	65.4 (-22.1)	49.5 (-17.1)
		✔				-	68.8 (-18.7)	52.5 (-14.1)
		✔	✔			-	69.1 (-18.4)	53.1 (-13.5)
		✔	✔	✔		✔	77.7 (-9.8)	58.4 (-8.2)
		✔	✔	✔	✔	✔	**82.0 (-5.5)**	**62.4 (-4.2)**
		✔	✔	✔	✔	✘	78.5 (-9.0)	59.5 (-7.1)
Teacher	-	-	-	-	-	-	87.5	66.6

Table 4.2: Ablation study of loss functions in our CFKD method with performance on OTB-100. GT means the student model is trained only with the groundtruth IoU labels. Multi-tasking means the final loss is the sum of the individual losses, and if crossed, it means that we first use $\mathcal{L}_{\text{fidelity}}$ and $\mathcal{L}_{\text{CF}}$ to distil the backbone, then use $\mathcal{L}_{\text{TS}}$ and $\mathcal{L}_{\text{AH}}$ to distil the IoUNet. The numbers are the AUC scores of the precision and success plots. The numbers in parentheses indicate the difference bewteen the teacher performance.

only 6% drop in both scores. This implies that the proposed CF loss indeed helps training of the small student backbone. Finally, when we distil the student backbone and IoUNet separately, i.e. firstly using fidelity and CF loss to distil the backbone, then using TS and AH loss to distil the IoUNet with the backbone parameters frozen, we see around 4% degradation in performance compared to the multi-task learning case (i.e. our CFKD method). The difference in the two cases is that when we combine all loss functions as in Equation 3.11, the errors produced by the losses for the IoUNet also contributes to the optimisation of backbone parameters, while in the other case they do not. The drop in performance when distilling the backbone and IoUNet shows that such flow of gradient is beneficial for the distillation of the entire network. Therefore, the results in Table 4.2 shows that each component in our proposed method is effective and contributes positively to the distillation process.

4.2.3 Backbone Architecture

To verify the effectiveness of using MobileNetV2 [83] as the student backbone architecture, we distil with CFKD method several students with different backbones and analyse their compression efficiency and tracking performance. Specifically, we three more students with backbones named ResNet-18-m (medium), ResNet-18-s (small), and ResNet-18-t (tiny). These backbones are all created by reducing the number of kernels in each layer of ResNet-18 by certain factor, which is 1/2 for medium, 1/4 for small, and 1/8 for tiny. The number of kernels in each layer of the IoUNets of these students are also reduced by the same ratio used for the backbone. The evaluation results of these networks are shown in Table 4.3.

It can be seen from Table 4.3 that the students with backbone ResNet-18-s and ResNet-18-t

	Backbone	Size	FLOPs	Backbone FPS	Tracking FPS	OTB-100 AUC (%)
Teacher	ResNet-18	109 MB	3.35 B	80 / 15	30 / 9	66.6
Student	ResNet-18-m	34 MB	0.87 B	90 / 25	35 / 11	57.1
	ResNet-18-s	8.7 MB	0.25 B	125 / 50	40 / 20	53.7
	ResNet-18-t	2.4 MB	0.073 B	130 / 60	45 / 25	48.3
	MobileNetV2	18 MB	0.33 B	80 / 40	38 / 20	62.4

Table 4.3: Evaluation of students with different backbone architectures obtained with our CFKD method. The speed is reported in typical frames per second (FPS) on GPU/CPU. The results of the ATOM and mATOM are also included for comparison.

have smaller model sizes and computational costs than our proposed student with MobileNetV2 backbone (mATOM), and they achieve higher frame rates both on GPU and on CPU. However, their performance is much worse than mATOM, with 14% and 23% degradation in AUC score on OTB-100 respectively compared to mATOM. On the other hand, the ResNet-18-m model whose model size is almost twice that of MobileNetV2 achieves 8% less AUC score than MobileNetV2 model while also tracking more slowly. This clearly indicates the advantage of the MobileNetV2, which uses depthwise and point-wise convolution to factorise the full convolution so as to reduce the computational cost significantly while still performing the filtering and combination operations. As a result, MobileNetV2 retains high capability while reducing the computations. These results show that our selected student architecture performs better than the manually created architectures, which is the way the competing distillation methods [63][94] create the student.

4.3 Comparison with Existing Distillation Methods

We compare our proposed method with existing knowledge distillation methods for trackers by distilling ATOM using these methods and evaluating the resulting trackers on OTB-100, TrackingNet and LaSOT datasets. The two methods we are comparing against are proposed by Wang et al. [94] and Liu et al. [63] which are both described in Section 2.4. In the following, we refer to these two methods by the name of the authors. Table 4.4 shows the results of distillation with these methods. In all three cases, the student architecture is our proposed MobileNetV2 model described in Section 3.2. This should not affect the performance of the two competing methods because they are not optimised for specific architectures.

From Table 4.3 we can see that on all test sets our method outperforms the other two in terms of all metrics. It is noted that the student ATOM tracker distilled with Wang et al.'s

		Wang et al. [94]	Liu et al. [63]	Ours
OTB-100	prec. (%)	12.5	74.2	**82.0**
	success (%)	10.0	55.1	**62.4**
LaSOT	norm. prec. (%)	4.3	43.3	**51.9**
	success (%)	5.2	38.4	**46.1**
TrackingNet	prec. (%)	11.0	56.9	**61.0**
	norm. prec. (%)	18.8	70.0	**74.2**
	success (%)	17.8	63.2	**66.9**

Table 4.4: Comparison of our method with the tracker distillation methods proposed by Wang et al. [94] and Liu et al. [63] on distilling the ATOM tracker. Prec. means precision, and norm. prec. stands for normalised precision.

method achieves extremely low scores on all datasets. This is because their method does not focus on teaching the student the feature representations produced by the teacher. Instead, their tracking loss (Equation 2.4) guides the student to learn features that aids conventional DCF-based tracking, which in ATOM is replaced by a CNN as described by Equation 1.3 in Section 1.4.2, instead of having the student learn generic features that can be utilised by the IoUNet and classifier of ATOM. On the other hand, our method uses the fidelity loss to directly force the student to produce similar features as the teacher.

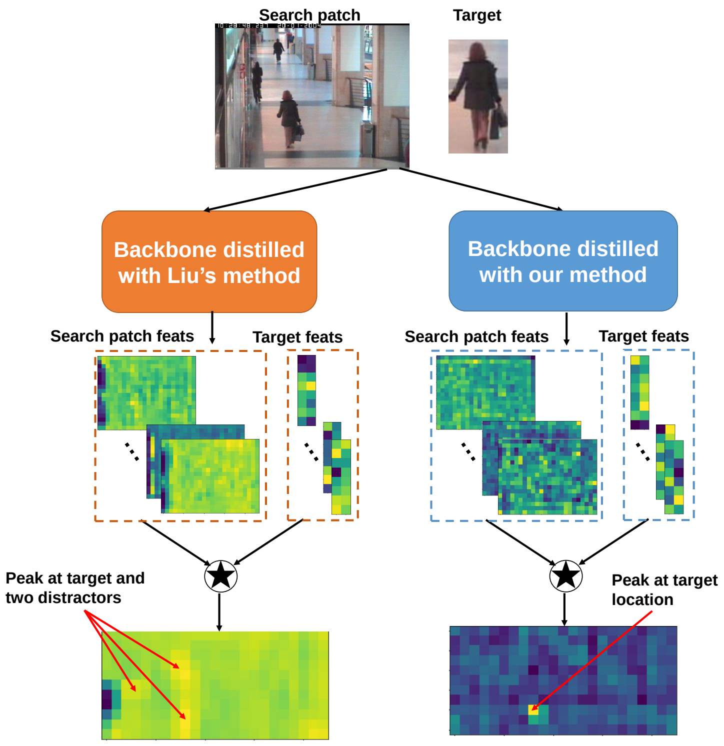
Search patch
Target
Backbone distilled with Liu's method
Backbone distilled with our method
Search patch feats
Target feats
Search patch feats
Target feats
Peak at target and two distractors
Peak at target location

Figure 4.1

Liu et al.'s method manages to train students with reasonably good performance, which however still has certain gap between the performance of the students trained with our method. This method also involves loss functions for matching the feature representations of backbones, but the features to match are weighted by a 2D map that has higher values at regions corresponding to the target (described in Equation 2.8 to 2.10), i.e. they put more focus on learning features for the target, so the resulting student may not show sufficient robustness to distractors. This is illustrated in Figure 4.1, which shows the "block4" features extracted from the target image and a search patch by students trained with Liu et al.'s method (on the left) and our method (on the right), as well as the cross-correlation output between these features. It can be seen that the cross-correlation output of the first student (left) has 3 peaks corresponding to the target and the two distractors in the search patch, while the output of the second student only has high values at the target location. This indicates that the feature extractor of the student trained with our method is more robust to distractors in the search patch.

4.4 Evaluation on DRNet

To show that our proposed distillation method generalises to other trackers from the ATOM family, we use it to distil the winner of VOT-2019 [50] challenge, DRNet. DRNet uses the same overall architecture and tracking algorithm as ATOM, but differs from ATOM in the bounding box regression step. While ATOM obtains the predicted box by optimising several proposals with gradient ascent to maximise their overlap with the target, DRNet directly predicts the bounding box coordinates with the IoUNet. Concretely, as shown in Figure 3.1, ATOM firstly uses the groundtruth bounding box in the reference frame and the features extracted from that frame to obtain a modulation vector, which is then fed into the IoU predictor along with the search patch features and bounding box proposals to predict the IoU score between the proposals and the target. DRNet, however, feeds the same set of inputs to the IoU predictor except the bounding box proposals, and its IoU predictor is modified to output the bounding box in the format of (x, y, w, h). This is illustrated in Figure 4.2 below, which is different from Figure 3.1 only in the input and output of the IoU predictor, and the backbone used (which is now ResNet-50).

Table 4.5 summarises the model reduction, speed improvement and performance on OTB-100 of the small DRNet named mDRNet (m for mobile) that is distilled with our proposed method, replacing the ResNet-50 backbone and the IoUNet with the student architecture described in Section 3.5. Note that while fidelity and CF losses are computed in the same way as before, the TS and AH losses are now taken between the 4-by-1 vector outputs (x, y, w, h) of the

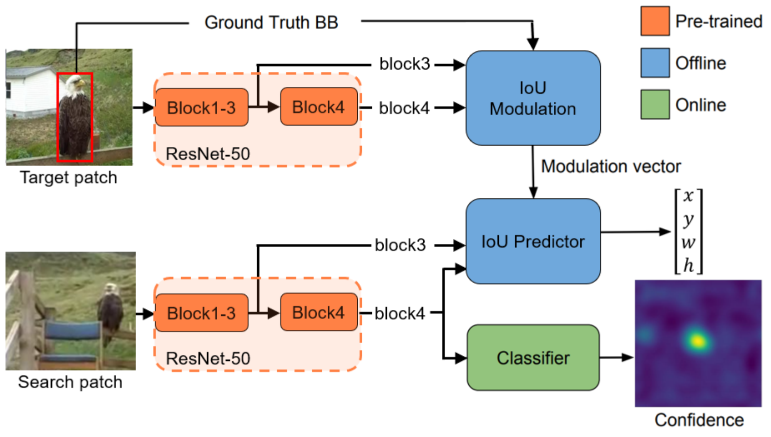

Figure 4.2: Overview of the DRNet tracker [50]. The differences from ATOM (Fig.3.1) are (i) the backbone architecture; (ii) the inputs and outputs of IoU predictor, which now directly outputs the coordinate, width and height of the predicted bounding box.

teacher and student IoUNets. The results for ATOM tracker are also included for comparison. Compared to the case of ATOM, our method reduces the size of DRNet and its computational costs more significantly (by around 95%) due to the heavier ResNet-50 of the teacher. The improvement in feature extraction speed on GPU is now more obvious than the ATOM case, with mDRNet extracting features at 80 frames per second compared to 50 FPS of teacher backbone. On CPU, the improvement is more significant (more than 4 times). In terms of tracking speed, mDRNet sees 50% enhancement on GPU and 200% increase on CPU compared to DRNet, achieving 26 FPS on CPU with less than 7% drop in performance on OTB-100, which is considered successful.

	Backbone	Size	FLOPs	Backbone FPS	Tracking FPS	OTB-100 AUC (%)
DRNet	ResNet-50	145.1 MB	7.84 B	50 / 9	40 / 12	67.6
mDRNet	MobileNetV2	18 MB	0.33 B	80 / 40	60 / 26	63.0
ATOM	ResNet-18	109 MB	3.35 B	80 / 15	30 / 9	66.6
mATOM	MobileNetV2	18 MB	0.33 B	80 / 40	38 / 20	62.4

Table 4.5: Results of distilling DRNet with our proposed method. The speed is reported in typical frames per second (FPS) on GPU/CPU. The results for ATOM are also included for comparison.

It is noted from Table 4.5 that despite the heavier backbone of DRNet, its tracking speed is higher than ATOM. This is because of the change in tracking mechanism from ATOM to DRNet described previously, i.e. DRNet does not involve the online gradient ascent step in ATOM for regressing the bounding box. Instead, DRNet directly predicts the box with one forward pass through the network. As a result, it tracks at higher frame rates in spite of its larger backbone.

We also observe that although both of the MobileNetV2-based trackers (mATOM and mDRNet) have the same overall architecture, mDRNet achieves higher AUC score than mATOM, which implies that the student trained with better-performing teacher (DRNet with 67.6% AUC) also achieves better performance.

4.5 Benchmarks Results

To verify that the trackers, mATOM and mDRNet, which are distilled with our method can achieve state-of-the-art performance while tracking in real time on CPUs, we evaluate them on popular benchmarks: OTB-100, LaSOT, TrackingNet, VOT-2018 and VOT-2019, and compare them with competing trackers on these datasets.

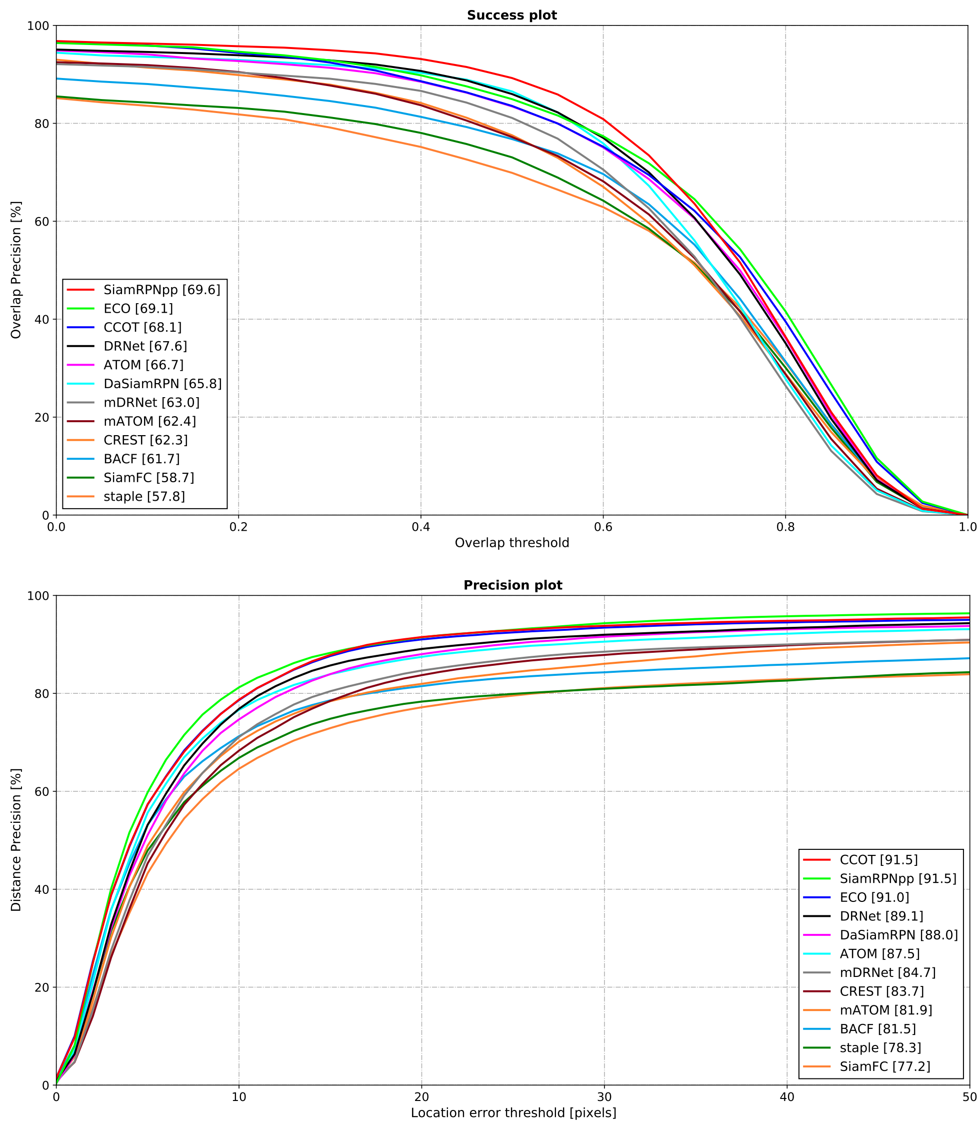

Figure 4.3: Success (top) and precision (bottom) plots with AUC of one-pass evaluation of state-of-the-art trackers on OTB-100.

Results on OTB-100. We compare mATOM and mDRNet with the best-performing trackers on OTB-100 including SiamRPN++ [54], ECO [20], CCOT [23], DaSiamRPN [104], CREST [86], BACF [46], SiamFC [8], Staple [7], as well as the two teacher trackers ATOM [19] and DRNet [50]. Figure 4.3 shows the success and precision plots of one-pass evaluation on OTB-100, with trackers ranked in terms of AUC scores in each plot. From the ranking in the success plot, which is the recommended primary metric suggested by the authors of the dataset, we see that there is certain performance gap between the distilled trackers and the best tracker (SiamRPN++) on OTB-100. However, such comparison is not fair since the teachers are already worse than SiamRPN++ on this dataset. Although several trackers rank higher than our distilled trackers, none of them can run real-time on CPUs. On the other hand, BACF and Staple are real-time trackers on CPU, but mATOM and mDRNet outperform them by approximately 8%.

	norm. prec. (%)	success (%)	prec. (%)
SiamRPNpp	59.7	52.0	51.3
ATOM	58.7	52.5	51.0
DRNet	57.6	48.5	46.7
mATOM	53.0	46.1	44.6
mDRNet	51.9	43.7	40.8
MDNet	47.6	41.4	38.2
VITAL	47.0	48.0	37.1
ROAM	46.0	40.6	38.4
SiamFC	43.6	34.9	34.6
StructSiam	43.2	34.7	34.1
SINT	36.9	32.7	30.2
ECO	35.0	33.7	31.0
ECO_hc	33.7	31.5	28.4

Table 4.6: Summary of state-of-the-art tracker results on LaSOT in terms of normalised precision, AUC, and precision scores.

Results on LaSOT. On LaSOT, we compare mATOM and mDRNet with their teachers ATOM, DRNet, and 9 other trackers including SiamRPN++ [54], MDNet [72], VITAL [87], ROAM [99], SiamFC [8], StructSiam [100], ECO [20] and SINT [89]. Figure 4.4 shows the success and precision plots of one-pass evaluation on LaSOT, where the trackers are ranked with their AUC scores on those plots. We observe that distilled with our method, both ATOM and DRNet see around 10% drop in the performance of their students on LaSOT. Despite such

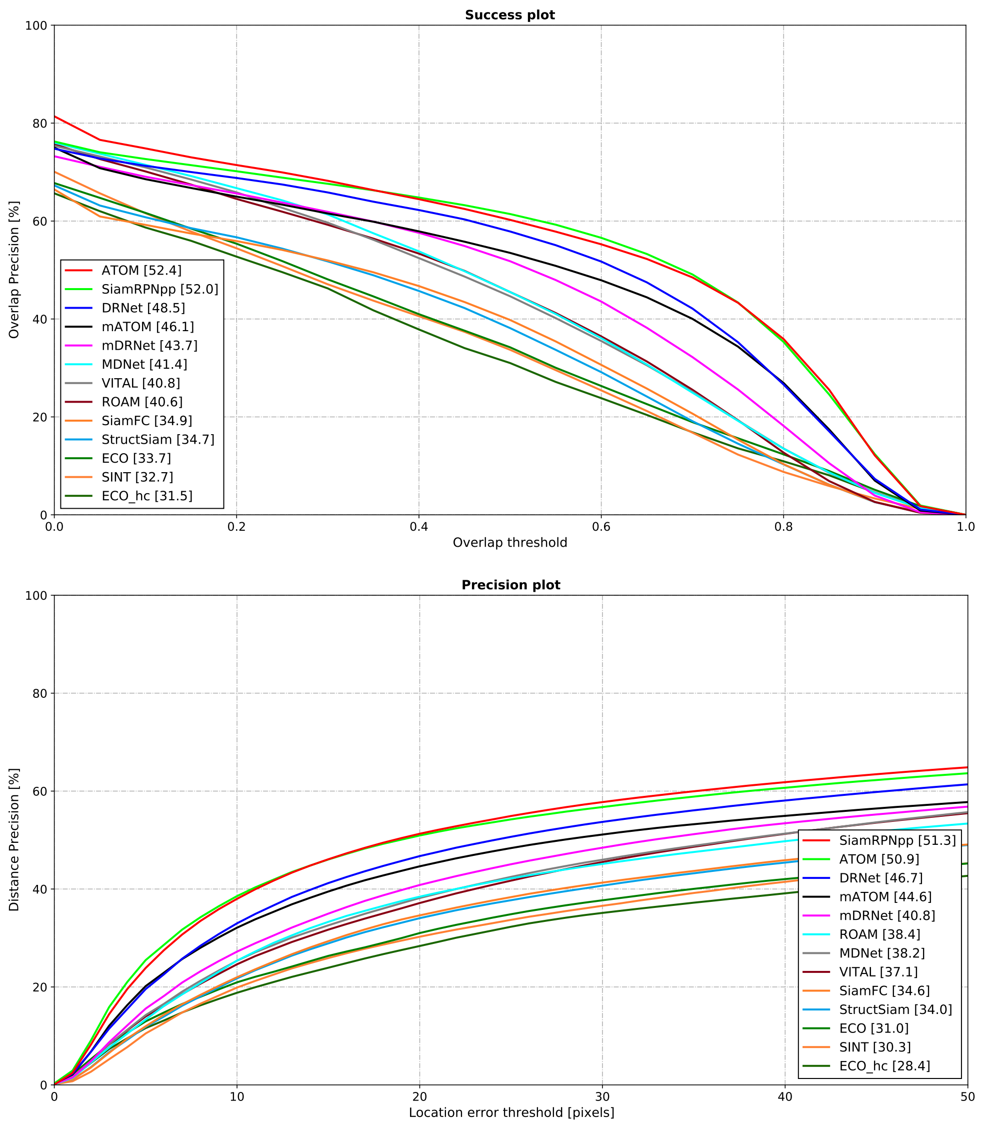

Figure 4.4: Success (top) and precision (bottom) plots with AUC of one-pass evaluation of state-of-the-art trackers on LaSOT.

drop, as CPU real-time trackers, these students still perform better than all competing trackers except SiamRPN++, which is real-time only on GPU. It is also noted that the ECO tracker that outperforms our students on OTB-100 performs 27% and 23% worse than mATOM and mDRNet respectively on LaSOT. We rank these trackers in terms of normalised precision and summarise the AUC and precision scores in Table 4.6. From the results on OTB-100 and LaSOT, we observe that the student trained with our method inherits the strength and weakness of the teacher, since the relative ranking bewteen mATOM and mDRNet on these two datasets follows that between ATOM and DRNet.

Results on TrackingNet. We compare our students and teachers with the best-performing trackers on TrackingNet including UPDT [9], MDNet [72], CFNet [90], SiamFC [8], DaSiamRPN [104], ECO [20], CSR-DCF [65] and Staple [7]. We observe that our student trackers outperform all other trackers in comparison despite the performance gap with their teachers while being able to track in real time on CPUs. Note that the other CPU real-time tracker Staple performs 21% worse than mATOM and 18% worse than mDRNet.

	success (%)	norm. prec. (%)	prec. (%)
ATOM	70.3	77.1	64.8
DRNet	69.4	77.2	64.0
mATOM	66.9	74.2	60.1
mDRNet	64.0	73.3	58.6
UPDT	61.1	70.2	55.7
MDNet	60.6	70.5	56.6
CFNet	57.8	65.4	53.3
SiamFC	67.1	66.6	53.3
DaSiamRPN	56.8	60.2	41.3
ECO	55.4	61.8	49.2
CSR-DCF	53.4	62.2	48.0
Staple	52.8	60.3	47.0

Table 4.7: Summary of state-of-the-art tracker results on TrackingNet, ranked by the AUC of the success plots.

VOT-2018 real-time			
	EAO	A	R
SiamRPN	0.383	0.586	0.276
mDRNet	0.364	0.556	0.183
mATOM	0.358	0.520	0.169
SA_Siam_R	0.337	0.566	0.258
SA_Siam_P	0.286	0.533	0.342
SiamVGG	0.275	0.531	0.337
CSRTPP	0.263	0.466	0.318
ATOM	0.245	0.553	0.194
DRNet	0.219	0.537	0.204

VOT-2019 real-time			
	EAO	A	R
SiamMargin	0.366	0.577	0.321
SiamFCOT	0.350	0.601	0.386
mDRNet	0.349	0.562	0.253
DiMP	0.321	0.582	0.371
DCFST	0.317	0.585	0.376
SiamDW_ST	0.299	0.600	0.467
mATOM	0.283	0.558	0.381
ATOM	0.240	0.596	0.557
DRNet	0.185	0.583	0.757

Table 4.8: Benchmark results on VOT-2018 (left) and VOT-2019 (right). Trackers are ranked in terms of EAO scores. A denotes accuracy and R is robustness.

Results on VOT-2018 and VOT-2019. For comparison on VOT challenges, we take the top-5 trackers from the VOT real-time challenge results provided in [48][50], and provide a summary in Table 4.8. During the real-time evaluation in VOT, frames are fed to the trackers at 20FPS, and if the trackers do not respond in time, their prediction from the last frame is used as the current prediction. And the reset-based mechanism is also used in the real-time evaluation. For consistency with the VOT protocol, the trackers are ranked in EAO, whose calculation takes into account both accuracy and robustness. It can be seen from Table 4.8 that under the real-time protocol, the teachers ATOM and DRNet cannot outperform their students anymore due to their slower tracking speed, which makes them lose track of many frames resulting in more failures which impairs the EAO score. This clearly demonstrates the effectiveness of our distillation method. Compared to the previous case where Siamese trackers dominate the real-time ranking, now with our distillation method we create DCF-based trackers with competitive real-time performance. In addition, although the Siamese trackers might be faster on GPUs, they cannot track in real-time on CPUs, which our DCF-based students are capable of.

Chapter 5

Conclusion and Future Work

In this project, we studied the problem of creating accurate and robust discriminative correlation filter-based trackers that can run real-time on CPUs, and proposed a solution called CFKD. Our solution is a knowledge distillation approach for creating compact neural models and benefits from cutting-edge work on CPU-efficient neural network architectures. We use MobileNetV2 as our student backbone architecture, and propose a combination of novel loss functions including the correlation filter loss for distilling ATOM, the state-of-the-art family of DCF trackers.

We performed extensive experiments on the ATOM tracker to study the impact of different components in our proposed method. We found that our student ATOM tracker can run more than two times faster than the teacher on a single-core CPU, tracking in real-time at 20 FPS. Our method was compared to two other existing knowledge distillation methods developed for trackers, and outperformed those methods. To study the generalisation ability of our method, we distilled DRNet and satisfactory results were obtained. We evaluated our distilled trackers on several popular benchmark datasets and compared them to state-of-the-art trackers. Our results show that without speed restriction, the students perform worse than the teachers in terms of accuracy, but such degradation is well-compensated by the significant improvement in tracking speed, especially on CPUs. Our results also showed that under speed restriction of 20 FPS, the student trackers outperform the teachers in overall accuracy and robustness. Therefore, we have proved our hypothesis that with appropriate knowledge distillation method, one can train a small network with similar performance to the teacher.

Although with our method one can distil trackers from ATOM family and obtain a CPU-real-time tracker, the compromise in offline tracking accuracy compared with the teacher networks might not be acceptable in certain applications, which is one of the weaknesses of our method. Another drawback of our method is the excessively long time (more than 24 hours) needed on GPUs for offline distillation, which might limit certain users that lack the sufficient computational resources. This is mainly due to cross-correlation operation during the calculation of the CF loss, which leads to significant more computations both in the forward and backward pass. While the second issue may be addressed with hardware acceleration, for the first problem of performance drop, we might need more sophisticated loss functions or distillation strategy to obtain more accurate real-time trackers. Our experiments did not cover modification of the MobileNetV2 to bring about more efficient networks at as little costs as possible. One of interesting future work can be to make use of the deep reinforcement learning architecture search algorithm introduced in [2] in order to obtain more efficient architecture for knowledge distillation.

Bibliography

[1] M. F. Abdelkader, R. Chellappa, Q. Zheng, and A. L. Chan. Integrated motion detection and tracking for visual surveillance. In *Fourth IEEE International Conference on Computer Vision Systems (ICVS'06)*, pages 28–28. IEEE, 2006.

[2] A. Ashok, N. Rhinehart, F. Beainy, and K. M. Kitani. N2n learning: Network to network compression via policy gradient reinforcement learning. *arXiv preprint arXiv:1709.06030*, 2017.

[3] S. Avidan. Support vector tracking. *IEEE transactions on pattern analysis and machine intelligence*, 26(8):1064–1072, 2004.

[4] S. Avidan. Ensemble tracking. *IEEE transactions on pattern analysis and machine intelligence*, 29(2):261–271, 2007.

[5] J. Ba and R. Caruana. Do deep nets really need to be deep? In *Advances in neural information processing systems*, pages 2654–2662, 2014.

[6] C. Bao, Y. Wu, H. Ling, and H. Ji. Real time robust l1 tracker using accelerated proximal gradient approach. In *2012 IEEE Conference on Computer Vision and Pattern Recognition*, pages 1830–1837. IEEE, 2012.

[7] L. Bertinetto, J. Valmadre, S. Golodetz, O. Miksik, and P. H. Torr. Staple: Complementary learners for real-time tracking. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1401–1409, 2016.

[8] L. Bertinetto, J. Valmadre, J. F. Henriques, A. Vedaldi, and P. H. Torr. Fully-convolutional siamese networks for object tracking. In *European conference on computer vision*, pages 850–865. Springer, 2016.

[9] G. Bhat, J. Johnander, M. Danelljan, F. Shahbaz Khan, and M. Felsberg. Unveiling the power of deep tracking. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 483–498, 2018.

[10] P. Bilinski, F. Bremond, and M. B. Kaaniche. Multiple object tracking with occlusions using hog descriptors and multi resolution images. In *3rd International Conference on Imaging for Crime Detection and Prevention (ICDP 2009)*, pages 1–6, 2009.

[11] D. S. Bolme, J. R. Beveridge, B. A. Draper, and Y. M. Lui. Visual object tracking using adaptive correlation filters. In *2010 IEEE computer society conference on computer vision and pattern recognition*, pages 2544–2550. IEEE, 2010.

[12] C. Buciluă, R. Caruana, and A. Niculescu-Mizil. Model compression. In *Proceedings of the 12th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 535–541, 2006.

[13] K. Chen and W. Tao. Once for all: A two-flow convolutional neural network for visual tracking. *IEEE Transactions on Circuits and Systems for Video Technology*, 28(12):3377–3386, 2018.

[14] T. Chen, I. Goodfellow, and J. Shlens. Net2net: Accelerating learning via knowledge transfer. *arXiv preprint arXiv:1511.05641*, 2015.

[15] Z. Chi, H. Li, H. Lu, and M.-H. Yang. Dual deep network for visual tracking. *IEEE Transactions on Image Processing*, 26(4):2005–2015, 2017.

[16] D. Comaniciu, V. Ramesh, and P. Meer. Kernel-based object tracking. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 25(5):564–577, 2003.

[17] D. Comaniciu, V. Ramesh, and P. Meer. Kernel-based object tracking. *IEEE Transactions on pattern analysis and machine intelligence*, 25(5):564–577, 2003.

[18] G. Cybenko. Approximation by superpositions of a sigmoidal function. *Mathematics of control, signals and systems*, 2(4):303–314, 1989.

[19] M. Danelljan, G. Bhat, F. S. Khan, and M. Felsberg. Atom: Accurate tracking by overlap maximization. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 4660–4669, 2019.

[20] M. Danelljan, G. Bhat, F. Shahbaz Khan, and M. Felsberg. Eco: Efficient convolution operators for tracking. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 6638–6646, 2017.

[21] M. Danelljan, G. Häger, F. Khan, and M. Felsberg. Accurate scale estimation for robust visual tracking. In *British Machine Vision Conference, Nottingham, September 1-5, 2014*. BMVA Press, 2014.

[22] M. Danelljan, G. Hager, F. Shahbaz Khan, and M. Felsberg. Learning spatially regularized correlation filters for visual tracking. In *Proceedings of the IEEE international conference on computer vision*, pages 4310–4318, 2015.

[23] 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*, pages 472–488. Springer, 2016.

[24] Y. N. Dauphin and Y. Bengio. Big neural networks waste capacity. *arXiv preprint arXiv:1301.3583*, 2013.

[25] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei. Imagenet: A large-scale hierarchical image database. In *2009 IEEE conference on computer vision and pattern recognition*, pages 248–255. Ieee, 2009.

[26] X. Dong and J. Shen. Triplet loss in siamese network for object tracking. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 459–474, 2018.

[27] B. A. Erol, A. Majumdar, J. Lwowski, P. Benavidez, P. Rad, and M. Jamshidi. Improved deep neural network object tracking system for applications in home robotics. In *Computational Intelligence for Pattern Recognition*, pages 369–395. Springer, 2018.

[28] 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. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 5374–5383, 2019.

[29] B. Funt, F. Ciurea, and J. McCann. Retinex in matlab. In *Color and Imaging Conference*, volume 2000, pages 112–121. Society for Imaging Science and Technology, 2000.

[30] R. Girshick. Fast r-cnn. In *Proceedings of the IEEE international conference on computer vision*, pages 1440–1448, 2015.

[31] H. Grabner, M. Grabner, and H. Bischof. Real-time tracking via on-line boosting. In *Bmvc*, volume 1, page 6, 2006.

[32] G. D. Hager and P. N. Belhumeur. Efficient region tracking with parametric models of geometry and illumination. *IEEE transactions on pattern analysis and machine intelligence*, 20(10):1025–1039, 1998.

[33] B. Han, J. Sim, and H. Adam. Branchout: Regularization for online ensemble tracking with convolutional neural networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 3356–3365, 2017.

[34] A. He, C. Luo, X. Tian, and W. Zeng. A twofold siamese network for real-time object tracking. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 4834–4843, 2018.

[35] K. He, G. Gkioxari, P. Dollár, and R. Girshick. Mask r-cnn. In *Proceedings of the IEEE international conference on computer vision*, pages 2961–2969, 2017.

[36] K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016.

[37] 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*, 37(3):583–596, 2014.

[38] G. Hinton, O. Vinyals, and J. Dean. Distilling the knowledge in a neural network. *arXiv preprint arXiv:1503.02531*, 2015.

[39] A. G. Howard, M. Zhu, B. Chen, D. Kalenichenko, W. Wang, T. Weyand, M. Andreetto, and H. Adam. Mobilenets: Efficient convolutional neural networks for mobile vision applications. *arXiv preprint arXiv:1704.04861*, 2017.

[40] C. Huang, S. Lucey, and D. Ramanan. Learning policies for adaptive tracking with deep feature cascades. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 105–114, 2017.

[41] A. D. Jepson, D. J. Fleet, and T. F. El-Maraghi. Robust online appearance models for visual tracking. *IEEE transactions on pattern analysis and machine intelligence*, 25(10):1296–1311, 2003.

[42] X. Jia, H. Lu, and M.-H. Yang. Visual tracking via adaptive structural local sparse appearance model. In *2012 IEEE Conference on computer vision and pattern recognition*, pages 1822–1829. IEEE, 2012.

[43] B. Jiang, R. Luo, J. Mao, T. Xiao, and Y. Jiang. Acquisition of localization confidence for accurate object detection. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 784–799, 2018.

[44] B. Jiang, R. Luo, J. Mao, T. Xiao, and Y. Jiang. Acquisition of localization confidence for accurate object detection. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 784–799, 2018.

[45] M. Ju and H. Kang. Illumination invariant face tracking and recognition. In *5th European Conference on Visual Media Production (CVMP 2008)*, pages 1–6, 2008.

[46] H. Kiani Galoogahi, A. Fagg, and S. Lucey. Learning background-aware correlation filters for visual tracking. In *Proceedings of the IEEE international conference on computer vision*, pages 1135–1143, 2017.

[47] D. P. Kingma and J. Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

[48] M. Kristan, A. Leonardis, J. Matas, M. Felsberg, R. Pflugfelder, L. Cehovin Zajc, T. Vojir, G. Bhat, A. Lukezic, A. Eldesokey, et al. The sixth visual object tracking vot2018 challenge results. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 0–0, 2018.

[49] M. Kristan, J. Matas, A. Leonardis, M. Felsberg, L. Cehovin, G. Fernandez, T. Vojir, G. Hager, G. Nebehay, and R. Pflugfelder. The visual object tracking vot2015 challenge results. In *Proceedings of the IEEE international conference on computer vision workshops*, pages 1–23, 2015.

[50] M. Kristan, J. Matas, A. Leonardis, M. Felsberg, R. Pflugfelder, J.-K. Kamarainen, L. Cehovin Zajc, O. Drbohlav, A. Lukezic, A. Berg, et al. The seventh visual object tracking vot2019 challenge results. In *Proceedings of the IEEE International Conference on Computer Vision Workshops*, pages 0–0, 2019.

[51] A. Krizhevsky, I. Sutskever, and G. E. Hinton. Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems*, pages 1097–1105, 2012.

[52] A. Krizhevsky, I. Sutskever, and G. E. Hinton. Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems*, pages 1097–1105, 2012.

[53] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998.

[54] B. Li, W. Wu, Q. Wang, F. Zhang, J. Xing, and J. Yan. Siamrpn++: Evolution of siamese visual tracking with very deep networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 4282–4291, 2019.

[55] B. Li, W. Wu, Q. Wang, F. Zhang, J. Xing, and J. Yan. Siamrpn++: Evolution of siamese visual tracking with very deep networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 4282–4291, 2019.

[56] B. Li, J. Yan, W. Wu, Z. Zhu, and X. Hu. High performance visual tracking with siamese region proposal network. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 8971–8980, 2018.

[57] B. Li, J. Yan, W. Wu, Z. Zhu, and X. Hu. High performance visual tracking with siamese region proposal network. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 8971–8980, 2018.

[58] P. Li, D. Wang, L. Wang, and H. Lu. Deep visual tracking: Review and experimental comparison. *Pattern Recognition*, 76:323–338, 2018.

[59] X. Li, W. Hu, Z. Zhang, X. Zhang, and G. Luo. Robust visual tracking based on incremental tensor subspace learning. In *2007 IEEE 11th International Conference on Computer Vision*, pages 1–8. IEEE, 2007.

[60] Y. Li, H. Ai, T. Yamashita, S. Lao, and M. Kawade. Tracking in low frame rate video: A cascade particle filter with discriminative observers of different life spans. *IEEE transactions on pattern analysis and machine intelligence*, 30(10):1728–1740, 2008.

[61] Y. Li and J. Zhu. A scale adaptive kernel correlation filter tracker with feature integration. In *European conference on computer vision*, pages 254–265. Springer, 2014.

[62] T.-Y. Lin, M. Maire, S. Belongie, J. Hays, P. Perona, D. Ramanan, P. Dollár, and C. L. Zitnick. Microsoft coco: Common objects in context. In *European conference on computer vision*, pages 740–755. Springer, 2014.

[63] Y. Liu, X. Dong, W. Wang, and J. Shen. Teacher-students knowledge distillation for siamese trackers. *arXiv preprint arXiv:1907.10586*, 2019.

[64] B. D. Lucas and T. Kanade. An iterative image registration technique with an application to stereo vision. In *Proceedings of the 7th International Joint Conference on Artificial Intelligence - Volume 2*, IJCAI'81, page 674–679, San Francisco, CA, USA, 1981. Morgan Kaufmann Publishers Inc.

[65] A. Lukezic, T. Vojir, L. Cehovin Zajc, J. Matas, and M. Kristan. Discriminative correlation filter with channel and spatial reliability. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 6309–6318, 2017.

[66] C. Ma, J.-B. Huang, X. Yang, and M.-H. Yang. Robust visual tracking via hierarchical convolutional features. *IEEE transactions on pattern analysis and machine intelligence*, 41(11):2709–2723, 2018.

[67] L. Matthews, T. Ishikawa, and S. Baker. The template update problem. *IEEE transactions on pattern analysis and machine intelligence*, 26(6):810–815, 2004.

[68] X. Mei and H. Ling. Robust visual tracking using $\ell 1$ minimization. In *2009 IEEE 12th international conference on computer vision*, pages 1436–1443. IEEE, 2009.

[69] X. Mei, H. Ling, Y. Wu, E. Blasch, and L. Bai. Minimum error bounded efficient $\ell 1$ tracker with occlusion detection. In *CVPR 2011*, pages 1257–1264. IEEE, 2011.

[70] M. Muller, A. Bibi, S. Giancola, S. Alsubaihi, and B. Ghanem. Trackingnet: A large-scale dataset and benchmark for object tracking in the wild. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 300–317, 2018.

[71] H. Nam, M. Baek, and B. Han. Modeling and propagating cnns in a tree structure for visual tracking. *arXiv preprint arXiv:1608.07242*, 2016.

[72] H. Nam and B. Han. Learning multi-domain convolutional neural networks for visual tracking. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 4293–4302, 2016.

[73] M. Orsic, I. Kreso, P. Bevandic, and S. Segvic. In defense of pre-trained imagenet architectures for real-time semantic segmentation of road-driving images. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 12607–12616, 2019.

[74] Y. Qi, S. Zhang, L. Qin, H. Yao, Q. Huang, J. Lim, and M.-H. Yang. Hedged deep tracking. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 4303–4311, 2016.

[75] J. Redmon, S. Divvala, R. Girshick, and A. Farhadi. You only look once: Unified, real-time object detection. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 779–788, 2016.

[76] 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*, pages 91–99, 2015.

[77] A. Romero, N. Ballas, S. E. Kahou, A. Chassang, C. Gatta, and Y. Bengio. Fitnets: Hints for thin deep nets. *arXiv preprint arXiv:1412.6550*, 2014.

[78] O. Ronneberger, P. Fischer, and T. Brox. U-net: Convolutional networks for biomedical image segmentation. In *International Conference on Medical image computing and computer-assisted intervention*, pages 234–241. Springer, 2015.

[79] D. A. Ross, J. Lim, R.-S. Lin, and M.-H. Yang. Incremental learning for robust visual tracking. *International journal of computer vision*, 77(1-3):125–141, 2008.

[80] D. E. Rumelhart, G. E. Hinton, and R. J. Williams. Learning representations by back-propagating errors. *nature*, 323(6088):533–536, 1986.

[81] A. Saffari, C. Leistner, J. Santner, M. Godec, and H. Bischof. On-line random forests. In *2009 ieee 12th international conference on computer vision workshops, iccv workshops*, pages 1393–1400. IEEE, 2009.

[82] I. Saleemi, L. Hartung, and M. Shah. Scene understanding by statistical modeling of motion patterns. In *2010 IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, pages 2069–2076. IEEE, 2010.

[83] M. Sandler, A. Howard, M. Zhu, A. Zhmoginov, and L.-C. Chen. Mobilenetv2: Inverted residuals and linear bottlenecks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 4510–4520, 2018.

[84] C. E. Shannon. A mathematical theory of communication. *The Bell system technical journal*, 27(3):379–423, 1948.

[85] K. Simonyan and A. Zisserman. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*, 2014.

[86] Y. Song, C. Ma, L. Gong, J. Zhang, R. W. Lau, and M.-H. Yang. Crest: Convolutional residual learning for visual tracking. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 2555–2564, 2017.

[87] Y. Song, C. Ma, X. Wu, L. Gong, L. Bao, W. Zuo, C. Shen, R. W. Lau, and M.-H. Yang. Vital: Visual tracking via adversarial learning. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 8990–8999, 2018.

[88] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich. Going deeper with convolutions. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1–9, 2015.

[89] R. Tao, E. Gavves, and A. W. Smeulders. Siamese instance search for tracking. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1420–1429, 2016.

[90] J. Valmadre, L. Bertinetto, J. Henriques, A. Vedaldi, and P. H. Torr. End-to-end representation learning for correlation filter based tracking. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 2805–2813, 2017.

[91] D. Wang, H. Lu, and M.-H. Yang. Online object tracking with sparse prototypes. *IEEE transactions on image processing*, 22(1):314–325, 2012.

[92] L. Wang, W. Ouyang, X. Wang, and H. Lu. Visual tracking with fully convolutional networks. In *Proceedings of the IEEE international conference on computer vision*, pages 3119–3127, 2015.

[93] N. Wang, J. Shi, D. Yeung, and J. Jia. Understanding and diagnosing visual tracking systems. In *2015 IEEE International Conference on Computer Vision (ICCV)*, pages 3101–3109, 2015.

[94] N. Wang, W. Zhou, Y. Song, C. Ma, and H. Li. Real-time correlation tracking via joint model compression and transfer. *IEEE Transactions on Image Processing*, 29:6123–6135, 2020.

[95] Q. Wang, L. Zhang, L. Bertinetto, W. Hu, and P. H. Torr. Fast online object tracking and segmentation: A unifying approach. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1328–1338, 2019.

[96] T. Wang, I. Y. Gu, and P. Shi. Object tracking using incremental 2d-pca learning and ml estimation. In *2007 IEEE International Conference on Acoustics, Speech and Signal Processing-ICASSP'07*, volume 1, pages I–933. IEEE, 2007.

[97] J. Wen, X. Li, X. Gao, and D. Tao. Incremental learning of weighted tensor subspace for visual tracking. In *2009 IEEE International Conference on Systems, Man and Cybernetics*, pages 3688–3693. IEEE, 2009.

[98] Y. Wu, J. Lim, and M.-H. Yang. Online object tracking: A benchmark. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2013.

[99] T. Yang, P. Xu, R. Hu, H. Chai, and A. B. Chan. Roam: Recurrently optimizing tracking model. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 6718–6727, 2020.

[100] Y. Zhang, L. Wang, J. Qi, D. Wang, M. Feng, and H. Lu. Structured siamese network for real-time visual tracking. In *Proceedings of the European conference on computer vision (ECCV)*, pages 351–366, 2018.

[101] W. Zhong, H. Lu, and M.-H. Yang. Robust object tracking via sparse collaborative appearance model. *IEEE Transactions on Image Processing*, 23(5):2356–2368, 2014.

[102] H. Zhou, Y. Yuan, and C. Shi. Object tracking using sift features and mean shift. *Computer vision and image understanding*, 113(3):345–352, 2009.

[103] G. Zhu, J. Wang, P. Wang, Y. Wu, and H. Lu. Feature distilled tracking. *IEEE transactions on cybernetics*, 49(2):440–452, 2017.

[104] Z. Zhu, Q. Wang, B. Li, W. Wu, J. Yan, and W. Hu. Distractor-aware siamese networks for visual object tracking. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 101–117, 2018.

[105] Z. Zhu, W. Wu, W. Zou, and J. Yan. End-to-end flow correlation tracking with spatial-temporal attention. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 548–557, 2018.