UNIVERSITÀ DEGLI STUDI DI MILANO

FACOLTÀ DI SCIENZE E TECNOLOGIE

Corso di Dottorato in Informatica - XXXIV Ciclo
Dipartimento di Informatica "Giovanni Degli Antoni"
Settore Scientifico Disciplinare INF/01

Advanced Methodologies for Visual Object Tracking

Emanuel Di Nardo

Tutor
Prof. Paolo Boldi

Co-Tutor
Prof. Angelo Ciaramella

Coordinatore del corso
Prof. Paolo Boldi

Anno Accademico 2020/2021

Abstract

Visual Object Tracking is a continuously developing and competitive field in computer vision and machine learning. It is based on the idea of being able to follow any object, unknown up to that moment, in its movements without losing it or letting it run away.

It is of fundamental importance to keep under observation a specific object otherwise difficult to track, in particular in very sensitive contexts such as security, more specifically in video surveillance, or autonomously guided or unmanned aerial vehicles (UAV), and in contexts in which it is of great help to perform automated tasks, such as in video production.

As in any other field, deep learning methodologies became part of visual tracking, bringing with them the state-of-the-art of many methodologies including object detection, super-resolution, adversarial generative networks, and image classification. Most of these methodologies cooperate in a single architecture dedicated to tracking, to the point that the computational resources required for this purpose are very intensive, having to coexist and collaborate with multiple deep models.

Visual object tracking has evolved more and more over the years from using correlation models based on transformations in the frequency domain, to allowing a global and highly efficient correlation, compatible, with learning based on gradient descent. It evolved again using models based on correlation of elements with similar characteristics extracted from a module that is able to recognize the characteristics of objects previously trained on the classification of images with large datasets. Moreover, we have recently witnessed the advent of transformers models with their explosion in the field of NLP and computer vision. These models were also rapidly included in methodologies for the field of visual tracking.

The following thesis aims to inspect methodologies that can enrich the current state-of-the-art by building and exploring architectures not yet defined in the literature as well as trying to improve those already developed.

We started our research with the idea of using Generative Adversarial Networks for their reconstruction power, putting the tracking problem in the point of view of subject segmentation.

Going further into the topic, the work continued with the application of models based on Siamese networks that effectively allow to correlate the element to be tracked with the research space itself. However, these models have already been widely studied which has led us to develop even more complex networks. Although we remained in the context of Siamese networks, we started from the use of conventional convolutional networks to novel techniques in computer vision as the Transformers. These have proven to be the center of interest of a large part of the scientific community in every field and, therefore, we tried to apply them in a field that is new compared to the state-of-the art. As a result we obtained a method that is able to be compared with all current techniques, obtaining scores that allow the implemented methodology, not only to enter in high positions in the benchmark leaderboards of the various datasets, but also to participate in VOT2022, which is the challenge of reference for the world of visual tracking and which is the goal of every tracking algorithm.

In addition, we investigated what the tracking algorithms are actually observing through methodologies of eXplainable Artificial Intelligence (XAI) and how the transformers and the attention mechanism play a very important role.

During the research activity learning models that are different from those based on traditional crisp logic have also been developed. The aim was to try to find a common point that could combine fuzzy logic with the flexibility of deep learning and how this can be used to explain the relationships between the data as the complexity of the neural network increases. Fuzzy logic was applied to transformers to build a Fuzzy Transformer, where the attention component is more easily explained given the success of fuzzy models in the field of explainability.

All the work of XAI has allowed us to verify how our idea of using the internal components of the mechanism of attention has produced a direct link between this and the elements on which we are actually going to act in order to produce the desired output. In addition, the experiments conducted on the fuzzy components, in this first phase, seems to validate our idea that not only can using a specific highly interpretable component produce results similar to the state of the art, but this also makes it easier to understand them.

Related Publications

Part of this thesis is published in:

- Di Nardo, E. (2019). Adversarial Learning for Visual Tracking Research Idea. In DDC@ AI* IA (pp. 101-106).
- Di Nardo, E., Ciaramella, A. (2021). Advanced Fuzzy Relational Neural Network. In WILF.

Results on *VOT2022 - Short Term - Bounding Box* track challenge have been accepted and they will published in:

- Kristan, Matej, et al. The tenth visual object tracking vot2022 challenge results. In European Conference on Computer Vision 2022.

Results are published on public benchmark dataset leaderboards:

- TrackingNet: http://eval.tracking-net.org/web/challenges/challenge-page/39/overview
- GOT10K: http://got-10k.aitestunion.com/leaderboard

The work has been submitted to *Information Sciences* journal.
Source code can be found on Github[1]

[1] https://github.com/EmanuelOverflow/ViTCRT

Table of contents

List of figures

List of tables

Introduction

To start, press any key.
Where's the ANY key?

Homer Simpson

In the field of Computer Vision, there are many tasks that are deeply present in everyday life. For instance, face recognition is used each time a photo is taken by a smartphone, as well as object classification is used to sort the photo album efficiently in order to provide a thorough search for users.

Nowadays, one of the most challenging tasks in computer vision is visual object tracking. In a nutshell, it can be defined as the procedure of locating an object of interest in a video sequence. Therefore, it is plain to see the importance given to this sector in terms of application.

For instance by considering the possibility of using it in the field of security and video surveillance where, it is possible to follow ill-intentioned people wandering in a sensitive place and verify they do not act illegally in the camera field of view.

Other uses could also be in the industrial sector, tracking of an object in the production process of a company or even tracking of a cell or a probe device in the biomedical field are a case in point. In addition, this can lead to the possibility of using tracking techniques in synergy with other artificial intelligence tools e.g. the classification of abnormal behavior.

As mentioned above, the difficulties that must be addressed come from its own characteristics that require having to deal with multiple issues simultaneously, ranging

from the classic ones found in computer vision, for instance changes in brightness or physical alteration of an object, to more complex ones such as the presence of similar objects within the image under analysis and the total or partial occlusion of an object that compromises in part or in its entirety its physical structure from different points of view.

Over time, several techniques have been studied to deal with each of these issues and gradually have made the Visual Object Tracking one of the most active fields of study. For this reason over the years new datasets [22, 23, 32, 57, 82] have been developed and continue to be developed to tackle the problem of visual tracking, trying to identify what the needs of the various methodologies are, e.g. having a sufficient or well characterized number of samples and have a homogeneous distribution of visual attributes.

All this leads to the creation of growing competition, so much so that some of these provide remote evaluation tools with leaderboards that allow to decide immediately the most efficient algorithm. Moreover, the competition continues with the introduction of real challenges, such as VOTChallenge [40, 41], which every year allows authors to compete in an exhaustive and well-balanced way, pushing them to make their work accessible and assessable. It all started by using methods based on almost punctual matching between two images [7, 45], based on features such as color, histograms or density functions [17]. Furthermore, techniques using dynamic models and a priori distribution of objects has been studied, as in the case of the Kalman filter [35], which assumes the existence of a discrete system described by linear equations that represent in turn a Gaussian distribution, or techniques based on Particle Filters [2] that are based on non-linear models where it takes into account the a posteriori probability of the elements. Still, it has been tried to model subspaces of features with techniques such as Principal Component Analysis (PCA) and Indipendent Component Analysis (ICA) [71]. Subsequently, other methodologies aimed to recognize which elements are background and which are not [3] using features such as Local Binary Pattern (LBP) [29], Histogram of Oriented Gradients (HOG) [53] and HAAR-like features [80]. It

was then decided to use global techniques that worked in the space of frequencies by exploiting the properties of convolution in the new domain, making these strategies much less sensitive to certain problems and consequently hardening the tracking [6].

The real turning point was with the advent of the use of artificial neural networks [59], which allowed to implement learning techniques with a high degree of generalization. In addition, it manually enabled to break away from the need to choose what the best features could have been by opting those of the objects that are the result of machine learning itself and that can be improved and adapted to the context from time to time.

This has paved the way new patterns of correlation, which did not take long to arrive with the use of the so-called Siamese networks [5]. These exploit the capabilities of convolutional networks allowing to relate the objects searched in the space where the search is made. All this has been the result of the constant innovation that has always taken place in the field of computer vision and artificial intelligence.

Nowadays, for instance, there is a discussion about neural architectures that can remember the long-term relationships between the elements that make up the data and find which are the relationships that best define them are. This does not only happen in computer vision, but also in other environments such as Natural Language Processing. However, the techniques used can be adapted to a wide-range of scenarios as well.

One of the proposed objectives is to use an architecture born for the NLP context and bring it in the context of images and yet adapt it to a new task different from the one initially proposed when using techniques already known in the state-of-the-art.

Another important aspect in any field of artificial intelligence and therefore also in the case of visual tracking too, is to be able to create models that are easily explainable through the use of eXplainable Artificial Intelligence (XAI) techniques. This field has become increasingly important over time to improve the confidence of humans in the results obtained. In this way the concept of black box that is often present in machine learning, where the results of the proposed solution are not explainable by the internal operations nor by its creators, is cancelled. To this end, various techniques

have been created, called post-hoc, because they allow you to explain a model from the output and the relationships that are internally generated, over the years, such as the technique of integrated gradients [77] that uses a simple technique of integration from the calculation of gradients produced by a neural network, or using DeepLIFT [76] that decomposes the response of a neural network for a specific input and propagates the response backward to each feature of the input.

In recent times with the rise of transformers we have also tried to interpret, instead, the response of the attention mechanism, to understand, if the neural network is effectively putting focus on the right elements. For this purpose it is possible to use algorithms such as the attention rollout [1] which gives the attention to all layers by using the information on residual connections to weigh the final attention matrix.

Another significant concept in the field of explainability also comes at the regulatory level, for example the European Union has introduced the "right to explanation" [2], in the GDPR. In this category, some Fuzzy-Rule based systems [8, 54, 55] are also having great importance, because they fall, instead, in the category of ante-hoc techniques, which is explainable by design. This has prompted the investigation on such topic, producing research to create alternative models that are easily explainable, and try improving existing techniques, for instance, attention mechanism, without losing the learning power generated in a neural network.

[2]Recital 71: https://www.privacy-regulation.eu/en/r71.htm

Chapter 1

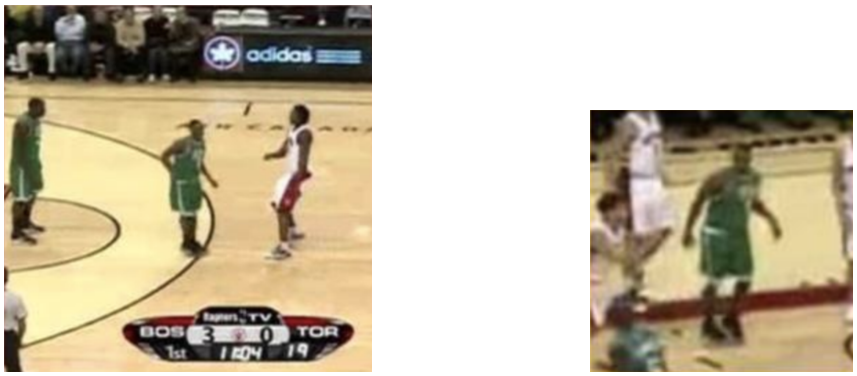
Visual Object Tracking

> Though this be madness, yet there is method in't.
>
> William Shakespeare

In this chapter the basis of Visual Object Tracking will be briefly explained, and in particular, the important difference between training and tracking phases of a tracker. Furthermore, visual attributes will be shown, with issues regarding every video sequence.

1.1 Introduction

Visual Object Tracking is the computer vision task where the proposed algorithm has to follow a designed object, which is called target or template z, in a video sequence, which is a sequence of frames. The single frame connected with the object is also called search image and it will be denoted by x (figure 1.1). This operation can happen by following the same criteria, based on the type of tracking that is needed. Mainly there are *Single Object Tracking* (SOT) and *Multiple Object Tracking* (MOT), where, as defined by the name itself, in the former there is an attempt to track a single object, while, in the latter, there are multiple objects of the same category. Tracking can



(a) (b)

Fig. 1.1 A search image at a specific time x_t in a video sequence and the target to follow specified at a previous time z_{t-1}

further be divided in some other categories, in [41] authors have identified in the Single Object Tracking two categories:

- **Short Term Tracking**: The target position is identified in each frame and the subject is always present in the sequence without ever disappearing
- **Long Term Tracking**: The target is not always visible in the scene and its position is not reported when the it is not visible in the frame.

Moreover, for those categories, as reported in [38], there are two sub-categories for each of them:

- *ST tracker* (ST_0): It is not required to re-detect the subject after it is lost and it is also not required to identify the occlusion type
- *ST tracker with conservative updates* (ST_1: The robustness of tracking process increases the update of the visual model based on a confidence estimation technique
- *Pseudo LT tracker* (LT_0): There is not a true re-detection method, it uses an internal method to identify and report tracking failure
- *Re-detecting LT tracker* (LT_1): The tracker is able to identify tracking failure and implements a proper re-detection algorithm.

Code	Attribute	Definition
CM	Camera Motion	Abrupt motion of the camera
IPR	In-Plane Rotation	The target rotates in the image plane
OPR	Out-Plane Rotation	The target rotates out of the image plane
DEF	Deformation	The target is deformable during tracking
FOC	Fully Occluded	The target is fully occluded in the sequence
POC	Partially Occluded	The target is partially occluded in the sequence
IV	Illumination Variation	The illumination in the target region changes
OV	Out-of-View	The target completely leaves the video frame
VC	Viewpoint Change	Viewpoint affects target appearance significantly
SV	Scale Variation	The ratio of bounding box is outside a range
BC	Background Clutters	The background has a similar appearance to the target
MB	Motion Blur	The target region is blurred due to target or camera motion
ARC	Aspect Ratio Change	The ratio of bounding box aspect ratio is outside a range
LR	Low Resolution	The target box is smaller than t_r pixels in at least one frame
FM	Fast Motion	The motion of the target is larger than the size of its bounding box
SOB	Similar Object	there are objects of similar shape or same type near the target

Table 1.1 Common visual attributes in video sequences

Another distinction that can be made is between model free and model based trackers:

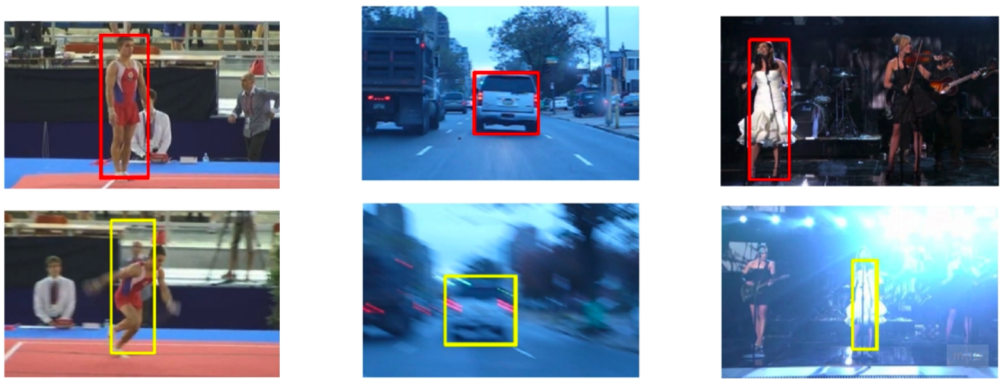
- *Model free tracking* aims to track an arbitrary object with no prior knowledge of the object itself. It means that the algorithm is not able to recognize if the target is a car, a person, a plane, or any other possible categorization. In this situation the methodology knows a generic characterization of any possible object and try to do its best to follow it and avoid to be distracted by similar objects

- *Model based tracking* is specialized on a specific object in the scene, for example, if it is needed to follow a face or a car. It has a thorough and specific knowledge of that object and cannot be used on others unless it has first been thoroughly retrained on that subject.

Typically, most research interests are on model free tracking that allow for a greater number of challenges while at the same time managing to produce excellent results with a significantly higher degree of generalization than their model based counterparts.

1.1.1 Visual Attributes

Tracking brings with it several issues that depend on the state of the target frame by frame. This means that the target changes throughout the video sequence and can



(a) (b) (c)

Fig. 1.2 Examples of visual attributes. (1.2a) Target deformation. (1.2b) Camera motion blur. (1.2c) Illumination and scale variation.

result in frequent errors in the tracker. These issues, which will be called *visual attribute*, are present, virtually, in every sequence and can combine with each other, among them it is possible to find: lighting changes, scale variations, occlusions, deformations, low resolution and many others that are reported in the table 1.1 with the associated codes used in the literature that work as a caption in section 3.4, while in the figure 1.2 there is a representation of some of them. This implies that the visual model of tracking must be able to be updated in order to cope with the continuous changes that the target may undergo with the possibility of locating it with the least number of errors.

1.2 Training and tracking phases

Despite what one might believe and what happens with other tasks both in the computer vision field and not, the training and testing/tracking phases differ from each other. In the training phase we usually choose to work with single images, not that there are no approaches based on video, but most of the known techniques prefer the first methodology. Partly, because in this way it is possible to create a method that works in real time and is therefore able to process frame by frame and does not need a greater number of frames to understand what is happening. One reason is that the amount of data available for training networks is never sufficient, so much so that many datasets

are used simultaneously to carry out training. Moreover, by using single frames it is also possible to propose targets that, even if they are part of the video sequence, are in different temporal moments and present alterations that are not directly present in the frame. This type of augmentation allows to reach higher generalization capacities than an approach based on direct matching. In the testing phase instead it is possible to add components to perform what is called *online learning*. This allows not to be strictly relegated to the target presented in the initialization phase, but to be able to learn how it transforms itself, as a result of movement and visual attributes, penalizing or encouraging the result, for example, of the confidence score obtained in the localization phase. It is also possible to add or turn off components later, such as the use of algorithms that reconstruct the segmentation mask of a given object.

Chapter 2

State of the Art in Single Object Tracking

> You do something and then someone else comes along and does it better
>
> *Pablo Picasso*

The chapter explains the state-of-the-art and modern methodologies for tracking. It starts with a general overview and then focus on most relevant methodologies that are also used or compared in the proposed work.

2.1 Introduction

Visual Object Tracking, is highly dependent on two inputs, the search area image and the target image. Dealing with this kind of problem is not an easy task, meaning that architectures have to be built to handle two data inputs that are different in size and significance, but are highly correlated. The key point is precisely this correlation that must take place in the most efficient way, both from the point of view of individual characteristics and from the point of view of the ability to correlate the two according to the chosen technique. One of the first effective methodologies that encapsulates this

idea is the Normalized Cross-Correlation filter (NCC) used in template matching [7], it is based on the assumption that two similar elements will have a high correlation score. Unfortunately, in tracking there are a lot of problems to face, such as, but not limited to, illumination and scale changes, occlusion, rotation, movement, blur and others. NCC is not able to deal with some of these problems which, by the way, are present in most real world scenarios.

Tracking has always attracted the interest of the scientific community and many other methodologies have been studied and implemented, but the most significant innovations have been the use of artificial neural networks and the transition from handcrafted, static features, to learning them from data itself. Obviously, this last change as well is due to the introduction of deep neural networks that are able to incredibly describe an object starting from pixels intensity values. Usually this family of methodologies come from *Image Classification* task.

The most important introductions are the *Siamese Neural Networks* [5] and the *Discriminative Correlation Filters* (DCFs) [19, 31]. The former is able to process two inputs and efficiently correlate them, the latter uses the properties of frequency domain to perform an effective operation by reducing the complexity of the model.

Over the last years a new category of architectures exploded in deep learning, they are called transformers [79], originally used to work on NLP problems, where they have reached state-of-the-art performance overwhelming the NLP sector. Recently they have also been using on image classification [20] and object detection [9] problems. Starting from those few examples they have shown great potential in tracking [12, 85] and the attention mechanism is a powerful alternative to cross-correlation modules. All those strategies are not limited their own architecture, they are modular and can be mixed together to build more robust and promising complex architectures.

2.2 Siamese Networks

State-of-the-art trackers are based on Siamese architecture [5]. After the introduction of Deep Neural Network in tracking, the usage of Siamese networks has been the greatest innovation in this field and they have been is use successfully for several years. The Siamese architecture uses the same model to process search and target images. It helps to extract features that are easy to correlate because the same feature extractor should extract characteristics that are very similar for similar objects. It means that a strong feature extractor can help to extract better results. Siamese networks are highly modular and a base structure is shown in figure 2.1. Four main modules can be identified with the following modules:

Backbone

Backbone is responsible for extracting features from the input images using neural networks that have a great discriminative ability, because they are aware of the main features of an object that has been previously learned. In the case where an item is totally new, they are still able to perform the best feature extrapolation that can be expected in an extreme case. This capability is made possible by a pre-training on the image classification task, usually using the ImageNet-1000 dataset [72]. That is, a thousand different classes of objects to be learned to recognize. The backbone can be fine tuned on the tracking task in the training phase. Most of the trackers in literature use ResNet50 as backbone [30]. Some tests are done also with a deeper ResNet, but it gives slightly better results at the cost of heavy computation, so the trade-off is not reasonable enough to prefer the most accurate versions.

Neck - Adjust Layer

Backbone outputs a high number of features, it depends on which internal layer is used to extract characteristics, for example using a ResNet50 is common to have 1024 (for internal *layer3*) and 2048 (for internal *layer4*) of them. Due to the high

data dimensionality *neck*, it is useful to down-sample the number of features, in order to work with a smaller number of components, coping with the process more easily, requiring less computational resources and keeping an high robustness. Neck is also shared for both inputs to preserve the property of the backbone where similar patterns have similar features. It will learn a non-linear sub-sampling projection from both data inputs.

Encoder - Body Layer

The *Encoder* or *Body* is the most interesting layer because it is responsible for all processing operations including correlation computation. It can be implemented to continue using the Siamese scheme or process the inputs separately to provide the network with the ability to learn a more specific representation of data. The second methodology remains very valid because it is influenced by the subsequent correlation that takes place between the two inputs. Therefore, the learning is conditioned by the gradients calculated on the operations carried out in the successive phases where the output component is the result of the search image processed using the features of the target image.

Head

The last part of the architecture is the *Head* module. Its role is to produce the desired output by mainly processing the correlated component resulting from the encoder part. Its use is not limited to the encoded data alone, but it is based on the task. It can use features coming from previous modules too, e.g. the backbone, the neck, or internal states of the encoder. The head can be divided in sub-modules or, as it is defined, in multiple heads, and each one is task-specific. There are three main sub-modules: bounding box regression, mask generation, foreground/background classification. These outputs are not all necessarily present at the same time.

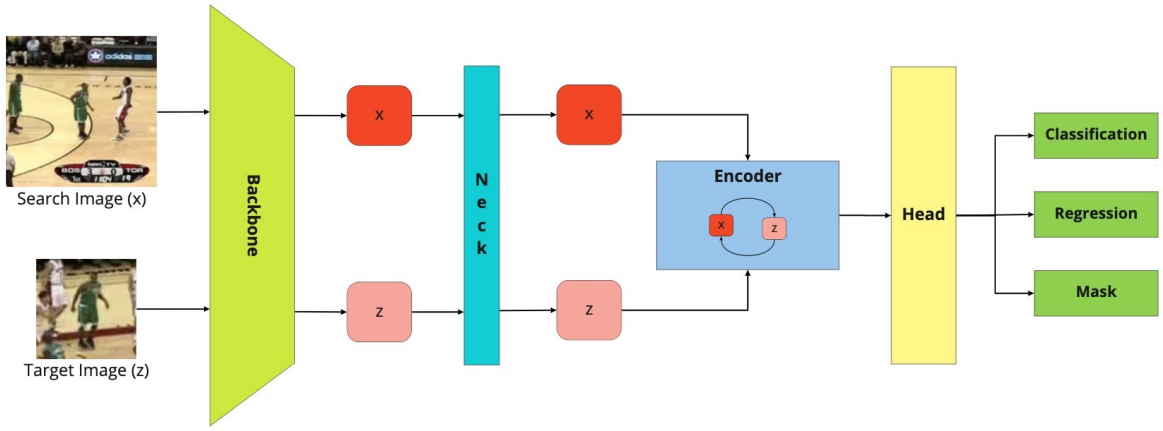


Fig. 2.1 Generic Siamese Architecture

Inspired by the literature the Siamese architecture is embedded in most of the proposed models. In chapter 4 an overview of each researched module will be presented without a specific architecture composition. In section 5.1 is shown how they have been organized in tracking architecture.

2.3 Head Selection

2.3.1 Bounding Box Regression

The main aim in visual object tracking is to find optimal images coordinates where it is possible to build the minimum area of the bounding box. It means that the object is centered in the bounding box and the edge of the box should perfectly enclose the target, avoiding including too many background components. Many techniques have been studied during the years starting from directly regress the two corners to use multiple areas where the target could be with a high probability. Generally, it is possible to divide these methods in two macro-categories:

- Anchors estimation;
- Anchor-free detection.

2.3.2 Region Proposal Networks

Firstly used in object detection *Region Proposal Network* (RPN) [66] performs greatly in locating objects. It works on multi-scale object detection introducing the concept of *anchors*. Each anchor can be seen as an *objects proposal* with an associated *objectiveness score*, which acts like a membership score, giving the probability that a region is of the class object identified by the network. The objects proposal and the object score are two components represented by two fully connected layers (figure 2.2): the box regression layer and the box classification layer, which share the same input corresponding to the spatial location in the feature map, producing two different, but linked responses. The two networks final layers are built using a number of features that are parametrized by the number of anchors k. The regression layer has $4k$ features to indicate the four coordinates of the bounding box corner for each anchor and the classification layer uses $2k$ features, which represent the prediction over the two classes (background/foreground) per anchor. Each output anchor can overlap with each other and a Non-Maxima Suppression (NMS) algorithm is used to suppress no maximum response based on the object score.

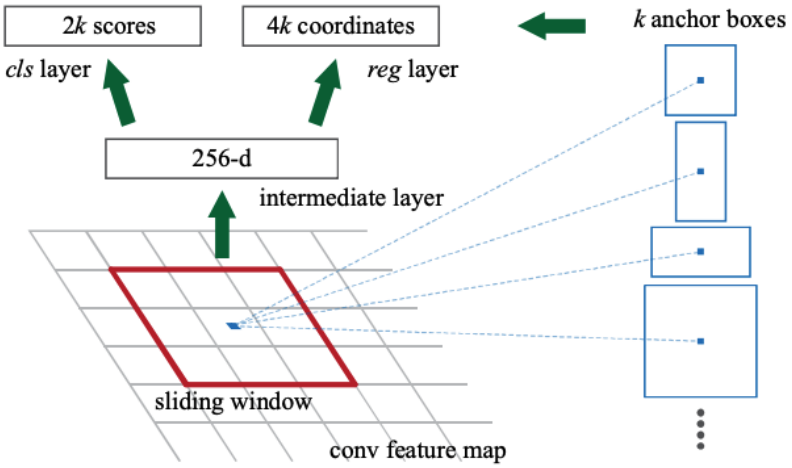


Fig. 2.2 Region Proposal Network structure schema

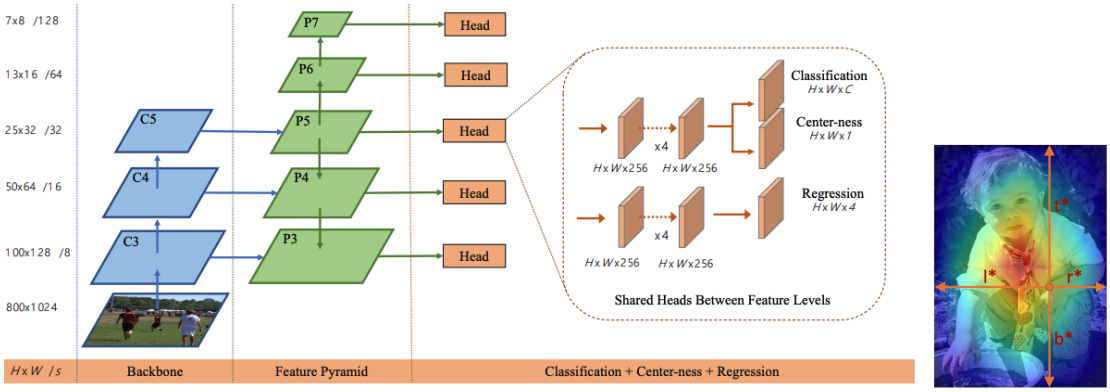
2.3.3 Fully Convolutional One-Stage Object Detection

Fully Convolutional One-Stage Object Detection (FCOS) [78] is an anchor-free object detector, meaning that it avoids the computation of hyperparameters needed to work with an anchor-based approach. It performs a bounding box estimation with $B_i = (x_0^i, y_0^i, x_1^i, y_1^1, c^i)$ where (x_0^i, y_0^i) are in the left-top corner coordinates of the bounding box, (x_1^i, y_1^i) are in the right-bottom corner coordinates and c^i is the class estimation, based on the number of classes in the dataset. Bounding boxes are found based on the (x, y) location on the last network feature map. It is considered a positive sample if it is located in the ground truth location and the class estimation is the expected one. The winning bounding box is the one with the minimum area. It uses two neural network layers at the top of the architectures, one for the bounding box regression and another for the box class. In FCOS the classification network does not predict the C classes, but uses C binary classifier for estimating (object / no-object) a score.

To ensure positive values for the regression branch an exp function is applied to the output vector. Multi-level estimation is achieved by using a Feature Pyramid Network approach [50] where feature maps extracted from multiple internal layers are used to compute multiple locations (figure 2.3a). The last important element in FCOS is the use of centerness (figure 2.3b) to determinate whether or not the prediction is valuable. It is computed by using the location target l^*, r^*, t^*, b^* where they corresponds respectively to [left, right, top, bottom] corners:

$$\sqrt{\frac{\min(l^*, r^*)}{\max(l^*, r^*)} \times \frac{\min(t^*, b^*)}{\max(t^*, b^*)}} \tag{2.1}$$

and is trained by using Binary Cross Entropy taking into account the regression output values. The aim is to weigh the classification score in order to get the prediction distance from the object center. The further it is, the more likely that the bounding box will be filtered out by the NMS.



(a) (b)

Fig. 2.3 (a) FCOS-FPN structure [93], with three output branch for each head. (b) Centerness where red color is equal to 1 and blue is equal to 0

2.3.4 Alpha-Refine

Alpha-Refine [86] is an anchor-free refinement module for tracking. As a tracking algorithm it aims to estimate the position of the target in the video sequence accurately. It adopts a Siamese strategy using a pixel-wise correlation strategy to preserve spatial location information. Most importantly, this has been designed to work as a refinement module, namely, it can be easily embedded in other trackers without the necessity of retraining or fine-tuning it on existing methodologies. It uses two branch heads, one for bounding box regression and an auxiliary one for mask generation. It uses a corner prediction with a set of (Convolution, Batch Normalization, ReLU) layers. The last layer is predicted by using a convolutional layer in order to predict two heatmaps, which are used for the top-left and bottom-right corners. In the end, a soft-argmax is applied to heatmaps allowing the precise localization of each component. An auxiliary mask head is also added for a more accurate estimation. It enhances the extraction of spatial information in a pixel-wise manner. The full tracker architecture is represented in figure 2.4.

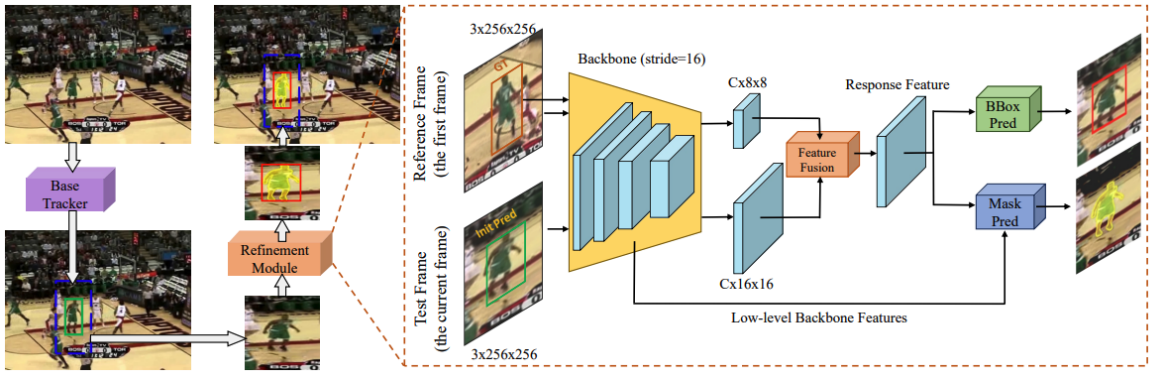


Fig. 2.4 Full tracking architecture of Alpha-Refine and the detail of refinement module [86]

2.4 Mask generation

Object segmentation mask is one of the possible outputs of a tracker. It can coexist and be trained with other heads. The most traditional module is composed by a sequence of convolutions of upsampling interpolations over the feature map (figure 2.5) [81]. Features extracted at any level of the backbone are involved to refine the shape of the object. Each feature maps in samples are downsampled from a high number of channels to a lower one, i.e. typically the first layer goes from 256 to 32. The same operation is performed on backbone features, bringing a number of features that is, as in the case of for the last component, from 2048 or 1024 to 32 (depending on the layer extraction). All the other features are scaled using the same processing schema. At each level the image features is added to the corresponding upsampling.

Another technique is to use a mask generation from semantic segmentation task [88], where the intra-class inconsistency is addressed by implementing a sequence of operations similar to the above methodology. However, in this case, different modules are use for refinement and low and high level features combination

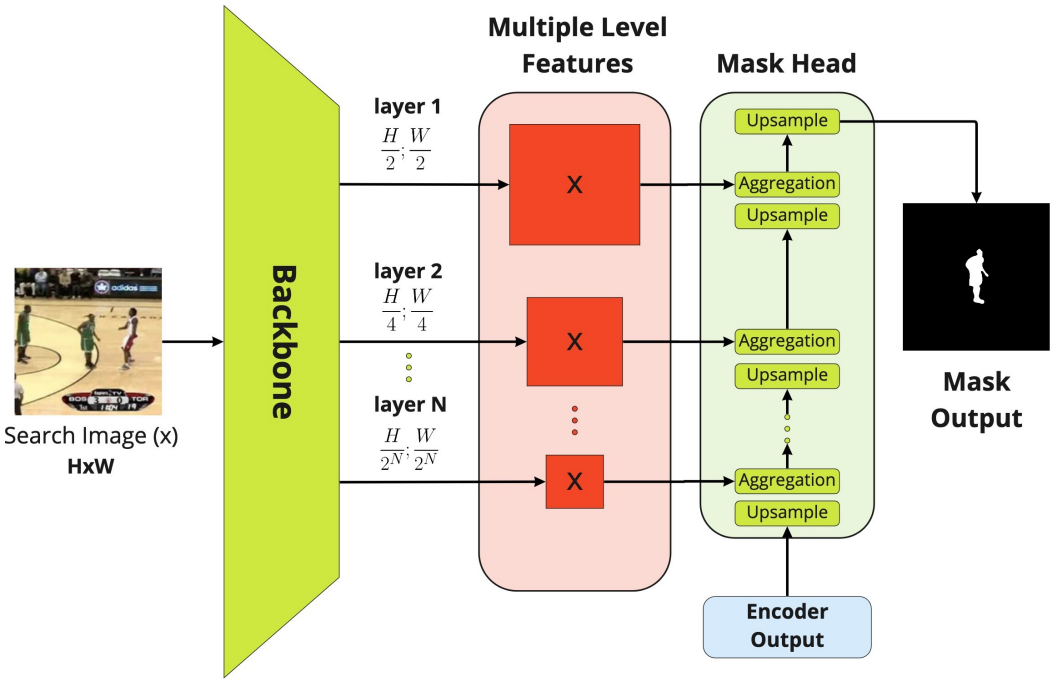


Fig. 2.5 Generic mask reconstruction head

2.5 Class estimation

Class prediction is the most simple branch. It assists the regression branch encouraging the regression to output the value in the same location where the classification found a foreground object. It usually outputs a map where each pixel corresponds to a foreground or background object as in [27, 93].

In score computation, the two methodologies match the index of the two classes and weighting the result or directly matching the ground truth map with the predicted one.

Chapter 3

Datasets and Metrics

> Why does a right angle measure ninety degrees? Misplaced question: it does not measure anything, it is others who measure him
>
> Umberto Eco

This chapter aims to introduce the datasets used for training and testing tracking algorithms, as well as the evaluation metrics used for performance comparisons. The chapter is divided in two main sections. In the first section the basic information and the visual attributes will be explained, representing the difficulties that a tracker has to face during tracking phase for each dataset. In the second section an overview of the metrics used in all datasets for evaluation will be addressed.

3.1 Introduction

Visual tracking is a hard problem to deal with. One of the most important elements is the possibility of having a good definition of the bounding boxes that encloses the object to search and that has an available number of modifications of the target that allows to capture all the important aspects of the real world. It must, therefore, be

able to provide real-world problem cases on which to perform training. Given the wide variance of objects that it is possible to track, hypothetically, every type of object, in its most generic meaning, is existent and non-existent at the same time. This gives us the opportunity to use not only a single dataset, but a set of them, so as to have an increasing number of elements for comparison. The training phase can vary from tracker to tracker: most of them prefer not to include temporal information in the training phase in order to reduce the dependence on the latter and to allow a greater generalization capability. The ability to track the same object over time can later be provided in the actual tracking phase through online learning strategies. A list of the main datasets used is given in the following paragraphs. A very important factor is to use a metric to compare trackers and also to compare performances between different datasets. As explained in the section 3.4 the most important ones are represented by the Area Under the characteristic Curve (AUC) [83], *Precision* and *Average Overlap* (AO) or the *Success* rate. In addition, most datasets provide benchmark tools to make sure the usage of the same metrics without errors. There are also datasets that are an integral part of competitions such as VOTChallenge [42] which provides its own metrics designed to compare the main features of the videos within it.

3.2 Datasets

Following a briefly presentation of datasets for training and benchmarking is provided below. It is reported how datasets are composed in term of number of frames and main features. A comparison on visual attributes is shown.

3.2.1 OTB

Online Object Tracking [82] provides 50 and 100 fully annotated sequences. It is divide in two versions the OTB50 (or OTB2013) and OTB100 (or OTB2015), the second is an expansion of the first version with more than 50 sequences. To evaluate the robustness of the trackers for common problems they are evaluated on more than 660.000 expected

Code	OTB	TrackingNet	LaSOT	GOT10K	VOT	NfS	TC128	UAV123
CM	✓	✓	✓		✓			✓
IPR	✓	✓	✓		✓		✓	
OPR	✓	✓	✓		✓		✓	
DEF	✓	✓	✓		✓	✓	✓	
FOC	✓	✓	✓	✓	✓	✓	✓	✓
POC	✓	✓	✓	✓	✓	✓	✓	✓
IV	✓	✓	✓	✓	✓	✓	✓	✓
OV	✓	✓	✓		✓	✓	✓	✓
VC			✓		✓	✓	✓	
SV	✓	✓	✓	✓	✓	✓	✓	✓
BC	✓	✓	✓		✓	✓	✓	✓
MB	✓	✓	✓		✓		✓	
ARC		✓	✓	✓	✓			✓
LR	✓	✓	✓	✓	✓	✓	✓	✓
FM	✓	✓	✓	✓	✓	✓	✓	✓
SOB		✓			✓			✓

Table 3.1 Visual attributes for main tracking datasets. The number of attributes can differ from those reported in the paragraphs due to the expansion of them in some dataset (i.e. *POC* and *FOC* in TrackingNet and LaSOT became *OCC* in OTB). Code caption can be found in 1.1.

bounding box. Video sequences used in evaluation have 11 different problematic visual attributes (table 3.1), many of them may be present at same time.

OTB laids the foundation for evaluation metrics *Success* and *Precision*. These metrics are explained in section 3.4

3.2.2 COCO

Common Object in Context (COCO) [49] is a large-scale object detection, segmentation, and captioning dataset released by Microsoft in 2014 for object recognition. It does not provide video sequences, but static labelled images. Objects are labeled using per-instance segmentations to aid in precise object localization. It contains 91 photos of objects types distributed in 328000 images with 2.5 million labeled instances. It contains 165482 training images, 81208 validation images and 81434 testing images with a high number of instances per category (figure 3.1). There is a very little chance of getting near-duplicate images across splits.

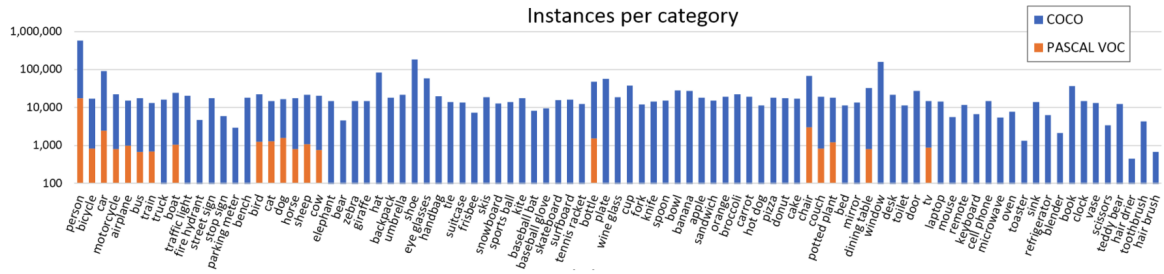


Fig. 3.1 Comparison of categories instances per category between COCO and Pascal VOC [21]

3.2.3 LaSOT

Large-scale Single Object Tracking (LaSOT) [22] is composed of 1.400 sequences with more than 3.5M frames which are Accurately annotated with a bounding box. The average video length of LaSOT is more than 2.500 frames. In each sequence a challenging scenario deriving from the wild is present, where target objects may disappear and re-appear again in the scene. It also contains 70 categories and most of them are a subset of ImageNet-1000 categories, with each consisting of twenty sequences. Long-term videos are provided in LaSOT and further bounding box specific rules are set for the same category. For example, in presence of mice, the tail is excluded from the bounding box because it gives an inconsistent deformation with respect to the object shape. As in OTB, a set of attributes have been designed to be associated with sequences. There are a total of 14 attributes (table 3.1), some of them are shared with OTB and TC-128 [47] datasets. This can be seen in figure 3.2a with the distribution over video instances. LaSOT introduced a new measure compared with the two introduced by OTB. It is called *Normalized Precision*, which is computed from the basic OTB version. This is explained in the section 3.4 with the other measures.

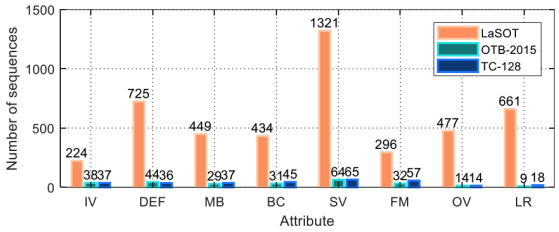
3.2.4 TrackingNet

TrackingNet [58] provides more than 30000 videos with more than 14 million dense bounding box annotations. It is released as a subset of Youtube-BoundingBoxes (YT-BB) dataset [65], which is designed for object detection with a number of video segments

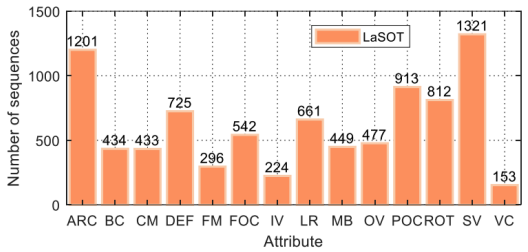
up to 380000, annotated at every second with upright bounding boxes. TrackingNet removed videos that do not present movements, such as potted plants and toilets, re-categorizing same classes to be better distributed based on subjects. In figure 3.3 the class distribution and the difference with YT-BB are showed. There are 30132 video sequences. The annotations are artificially provided by a DCF tracker [31]. The test set is composed of 511 videos that are labelled using Amazon Mechanical Turk service. Furthermore, in TrackingNet there are 15 visual attributes (table 3.1). The evaluation is performed on an evaluation server and can not be done locally.

3.2.5 GOT10K

Generic Object Tracking in the Wild [32] is built upon the backbone of WordNet structure [56] and it populates the majority of over 560 classes of moving objects and 87 motion patterns. It provides more than 10.000 video segments with more than 1.5 million manually labeled bounding boxes. Test sets that consist of 420 videos belong to 84 object classes and 31 motion classes. In addition, the one-shot protocol for tracker evaluation is introduced, where there is no overlapping of classes in training and test. The video collection started using five nouns from WordNet: animal, person, artifact, natural object. For motion classes instead, the authors expanded their search including: locomotion, action, and sport. Word sub-trees are filtered and pruned (e.g. removing extinct, static and repeated object classes, grouping close sub-classes, etc.). 2.500 object classes and over 100 motion classes have been obtained in this first phase. To improve the efficacy, 2.500 object classes are categorized into 121 groups with the



(a) LaSOT distribution comparison for common attributes



(b) LaSOT distribution of sequences for each attribute

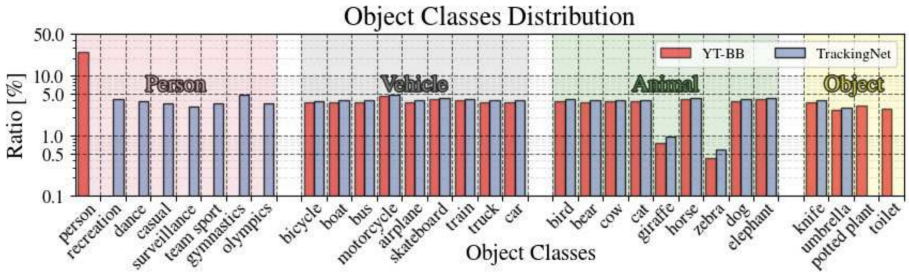


Fig. 3.3 Class distribution and aggregation with YT-BB differences.

purpose of making sure that each one is collected. Then, we rank the object classes in each group based on their corresponding searching volumes on the YouTube website. After many verification phases the final pool contains 563 classes of moving objects and 87 classes of motion. The evaluation protocol requires that trackers must be trained only on the training set provided by GOT10K, without using datasets from other sources. Also in this case the evaluation is performed on an evaluation server and can not be done locally.

3.2.6 NfS

Need for Speed [23] is a higher frame rate video dataset consisting in 100 videos, for a total of 380k frames with a frame rate of 240 FPS. 75 sequences are captured using an iPhone and an iPad Pro, and the remaining 25 are taken from Youtube. Each frame is annotated with an axis aligned bounding box and labeled with nine visual attributes 3.1.

3.2.7 TC128

Temple Color 128 [47] is based on the idea that color information is important in computer vision tasks and most of all in the tracking task. It is composed of 128 color sequences annotated with visual attributes 3.1.

3.2.8 UAV123

UAV123 [57] is an aerial video dataset for low altitude UAV tracking. It provides 123 fully annotated HD video sequences with more than 110k frames. Each video is annotated with many attributes 3.1. The dataset is composed of videos captured with three different devices. 103 sequences have been captured using a professional-grade UAV (DJI S1000) with an altitude varying between 5 to 25 meters and a variable framerate in a range between $30 - 96$. 12 sequences have been captured using a boardcam, without image stabilization, mounted on an UAV. The last 8 sequences are synthetic and produced by the UAV simulator provided by the authors.

3.2.9 VOT

Many versions of *Visual Object Tracking* (VOT) dataset exist. Approximately, each year a new version of the dataset is released. It is a compound of videos in many other dataset test sets (OTB, GOT10k, NfS, etc.). It can not be used for training, but it is a challenging dataset that is coupled with VOT Challenge (section 3.3). In VOT2013, targets were annotated by axis-aligned bounding boxes; in VOT2014 rotated bounding boxes are used; from VOT2020 to VOT2021, bounding boxes are not longer in use, as a precise segmentation mask is preferred. The current edition of the VOT2022 reintroduces the Bounding Box sub-challenge in the short-term tracking.

3.3 VOT Challenge

VOT [38–40] is an annual challenge where the best trackers challenge each other on many tasks, there are five challenges:

1. **VOT-ST**: for short term tracking. Starting from VOT22 it has been divided into two other sub-categories:
   (a) *Segmentation*: trackers must provide a segmentation mask of the object
   (b) *Bounding Box*: trackers must provide the enclosing bounding box

2. **VOT-RT**: a short-term tracking that requires real time performances
3. **VOT-LT**: for long-term tracking
4. **VOT-RGBT**: it is a short term tracking with thermal imagery
5. **VOT-RGBD**: a long term tracking with depth dimension

The evaluation of VOT-ST and VOT-RT is on segmentation mask. Instead, the rest requires bounding box output. The organizers give an evaluation toolkit that help to download the sequences and perform the tracking by using *TraX* [10] a protocol that communicates with the tracker code to send frames, it then receives the appropriate output that will be processed by the toolkit. In the challenge, it is prohibited to train a tracker on the following datasets: OTB, VOT, ALOV, UAV123, NUSPRO, TempleColor, RGBT234 and 1000 sequences in GOT10K. This happens because many sequences in VOT come from these datasets. Furthermore, the use of class labels in not allowed.

3.4 Metrics

Given the wide variety of datasets present, it is also necessary to define the metrics used to define the capability of a tracking algorithm. Among those presented LaSOT and TrackingNet are based on what OTB introduced about the calculation of Success, Accuracy and Normalized Accuracy. The *One-Pass Evaluation* (OPE) methodology is used, i.e., the tracker is initialized with the first bounding box and lets it proceed with tracking. The three metrics are calculated as follows:

- **Success (S)** is computed as the Intersection Over Union of the pixels of the ground truth bounding box b_{gt} and the tracker b_t output. It measures the bounding box overlap score and is computed as:

$$S = \frac{|b_{gt} \cap b_t|}{|b_{gt} \cup b_t|} \tag{3.1}$$

where $\bigcap$ is the intersection and $\bigcup$ is the union of the two regions denoted by $|.|$, that is the number of pixels in the region. To measure the performance on a sequence of frames, the number of successful frames whose overlap S is larger than the given threshold $t_o \in [0, 1]$ is counted. The success rate is not considered at a certain t_o, but the corresponding AUC is evaluated.

- **Precision (P)** is the center location error, which is defined as the Euclidean distance between the center locations of the tracked targets c_t and the ground truths c_{gt}. To overcome the tracker's problem of identifying a region that is far from the true target, a distance threshold is taken into account (usually equals to 20 pixels):

$$P = ||c_t - c_{gt}||_2 \tag{3.2}$$

- **Normalized Precision (P_{norm})** due to the fact that precision P is sensitive to the image resolution, the normalized version is used to remove the dependency on the image size:

$$P_{norm} = ||W(c_t - c_{gt})||_2 \quad \text{where} \quad W = diag(b^x_{gt}, b^y_{gt}) \tag{3.3}$$

Unfortunately, not all datasets use the same metrics. VOT uses its own metrics [38, 41] to calculate three key elements: *Expected Average Overlap* (EAO), *Robustness* (R) and the *Accuracy* (A). VOT needs the concept of *anchors* and *failure*. The former is an initialization point in the sequence and is placed in a specific point in order to reduce the bias of re-initialization. It divides a sequence in sub-sequences with a specific starting point of a of length $\Delta_{anc} = 50$. The latter happens when the overlap of the predicted bounding box b_t over the ground truth b_{gt} is less than a threshold value θ_t and tracker does not recover the target in a specific window time of $\theta_N = 10$ frames. After a failure the tracker must be re-initialized. This concept can introduce a bias in the tracking because a failure at time t can also occur at time t_{+1} since it is easy to

assume that in the next frame the type of visual attribute creating the problem can persist. To avoid it a number of frames equals to $N_{skip} = 5$ are skipped. This notion is useful for short-term tracking where the length of the video is short and penalize those which fail frequently. All measures are computed on sub-sequences:

- **Accuracy**: taking into account a sequence s that starts from an anchor a, the accuracy $A_{s,a}$ is the average overlap between the target prediction and the ground truth computed on the frames before the tracker failure:

$$A_{s,a} = \frac{1}{N^F_{s,a}} \sum_{i=1:N^F_{s,a}} \Omega_{s,a}(i) \tag{3.4}$$

  $N^F_{s,a}$ is the number of frames before a failure and $\Omega_{s,a}(i)$ is the overlap between the prediction and the ground truth at frame i. The accuracy for a sequence is the average value of all sub-sequences accuracy:

$$A_s = \frac{1}{\sum_{a=1:N^A_s} N^F_{s,a}} \sum_{a=1:N^A_s} A_{s,a} N^F_{s,a} \tag{3.5}$$

  where N^A_s is the number of anchors in the sequence s. The total accuracy A is the average of all sequences accuracy:

$$A = \frac{1}{\sum_{s=1:N} N^F_s} \sum_{s=1:N} A_s N^F_s \tag{3.6}$$

  where N is the number of sequences in the dataset, N_s is the number of frames in sequence s and $N^F_s = \sum_{a=1:N^A_s} A_{s,a} N^F_{s,a}$ is the number of frames used to compute the accuracy.

- **Robustness**: $R_{s,a}$ is defined as the number of times the tracker fails on sub-sequence:

$$R_{s,a} = \frac{N^F_{s,a}}{N_{s,a}} \tag{3.7}$$

as in accuracy $N_{s,a}^{F}$ is the number of frames before a failure and $N_{s,a}$ is the number of frames in a sub-sequence. Robustness on a sequence R_s is the average of all sub-sequence robustness:

$$R_s = \frac{1}{\sum_{a=1:N_s^A} N_{s,a}} \sum_{a=1:N_s^A} R_{s,a} N_{s,a} \tag{3.8}$$

where N_s^A is the number of anchors in the sequence s. The total robustness R is the average of all sequences robustness:

$$R = \frac{1}{\sum_{s=1:N} N_s} \sum_{s=1:N} R_s N_s \tag{3.9}$$

- **Expected Average Overlap**: is the combination of accuracy and robustness. The EAO curve is computed and averaged over a typical short-term sequence interval. If a tracker fails in a sub-sequence (s, a) the overlap drop to zero. The EAO curve $\hat{\Phi}_i$ at sequence length i is defined as:

$$\hat{\Phi}_i = \frac{1}{|\mathcal{S}(i)|} \cdot \sum_{s,a \in \mathcal{S}(i)} \Phi_{s,a}(i) \tag{3.10}$$

where $\Phi_{s,a}(i)$ is the average overlap calculated between the first frame and the i-th frame of a sub-sequence starting at anchor a, $\mathcal{S}(i)$ is the set of sub-sequences with length greater or equal to i; $|\mathcal{S}(i)|$ is the number of the sub-sequences. The curve is then computed in a interval bounds $[N_{low}, N_{hi}]$ determined by the mean $\pm$, one standard deviation of the anchor generated sub-sequences.

Chapter 4

Proposed methodologies

The best ideas don't come from reason, but from lucid, visionary folly

Desiderius Erasmus Roterodamus

In the following chapter the methodologies are described. These consist of state-of-art techniques and new proposed solutions acting as building blocks of a series of cutting-edge tracking strategies. The chapter is organized starting from preliminary studies on Generative Adversarial Networks to methodologies that are used in literature, such as Siamese networks and online learning.

4.1 Introduction

The main purpose of the research activity is to give an alternative strategy to visual object tracking by introducing new ideas to study the feasibility and possibility of presenting some techniques not adopted nowadays. The study started from baseline ideas of representing the alteration of the target during the training phase and reducing the influence of them during the tracking phase. During the research activity, many ways have been tested, with the influence of the state-of-the-art methodologies coming from several contexts, not just from computer vision, but also, for example, from

Natural Language Processing. Hundreds of architectures have been developed and tested to discover a valid guideline and limitations.

4.2 Generative Adversarial Networks

Generative Adversarial Networks (GANs) have the ability to generate fake data directly derived from the capacity to approximate the input data distribution, in this way, we can perform various tasks, such as image generation, denoising, super-resolution, in-painting, and so on. Incredible results have been reported with regards to those tasks, so the idea is: why not try using the generative capacity of the network to estimate the position of a target in an image? And if it works, is the latent space, learned from subjects, stable and not greatly influenced by variations and deformations? Usually, GANs are very effective on a single task generation. The entire network, for example, is trained on *face generation* or *face translation*. Alternatively, there is a strong tendency of being trained on *semantic segmentation* or *picture in-painting* because the objective of the network is to learn a common latent space that is able to map all elements of a specific nature and not more than one.

Proposed modification

Tracking is a more complex problem from this point of view, because there is not a specific object to map, but the network needs to learn a generalization of multiple and possibly unknown subjects. To face this problem, the idea is to simplify the problem by going from multiple objects to simply learn an input object that is strictly related to the image as a background/foreground problem. This brings us to the main notion of giving to a generative adversarial architecture the input image and the target as inputs to the generator, getting the resulting foreground mask reconstruction in output. The discriminator objective is commonly used to understand if the input foreground mask is the output of the generator or the ground-true. From the mask output a bounding-box is generated to recognize the target spatially.

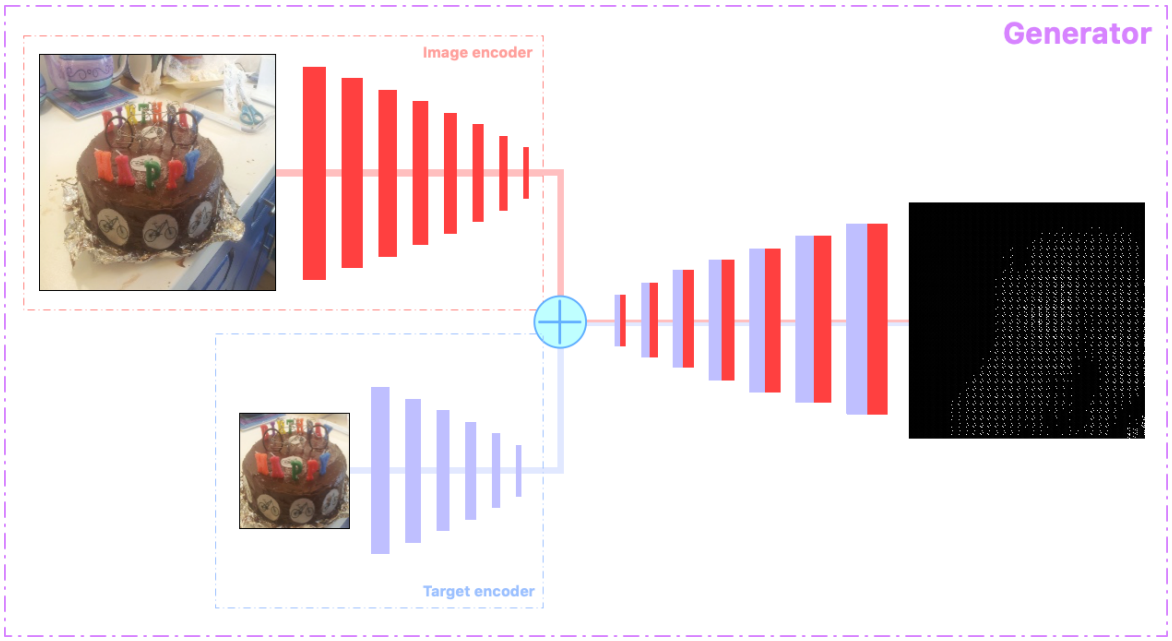


Fig. 4.1 Correlation DCGAN

4.2.1 Proposed strategy: Correlation DCGAN

The first attempt is to put into practice the concept of a *Deep Convolutional GAN* to purely work with images without restructuring them. In the naive GAN random noise generated from a normal distribution is used as input tensor to generate a random and coherent output. In the tracking case, it is mandatory to give information on the search image and the target image. These have been utilized as input for a GAN with a two-branch input, without a direct interaction. Inspired by the correlation strategy in [5], a cross-correlation is carried out on the last layer of the Generator Encoder. A depthwise convolution is used taking as input the search image feature maps x_{enc}^{L-1} and the target feature maps z_{enc}^{L-1} as convolutional weights. It is guaranteed that the target feature maps are correlated over all channels in the search image, getting the maximum correlation factor on depth dimension. The single output correlation is performed in the next step as input for the Generator decoder in order to reconstruct an image with same size as the original, namely, the resulting foreground mask of the correlation of both inputs. During the reconstruction phase a *skip-connection* is used as in [70], where the features are extracted from the search image at any level

x^l_{enc} where $l \in [0, L-1]$ are concatenated with the decoded one, the idea is to reinforce the correlation of the target that is present in the image to obtain a fine-reconstruction of the subject. In figure 4.1 it is possible to observe the network structure. In order to better understand the capabilities of the GAN and to get a good training many architectures have been tested. The second attempt is to modify the encoder structure trying to give multiple feedback by correlation to image search. At each encoder step, this is developed through the execution of the correlation by target depthwise convolution over image, propagating forward the result of the correlation as search image features map. In this process, the current target does not have any computation other than convolutional encoder layer. It is designed to increase the correlation score at each layer with an explicit relation to learn in the backpropagation phase automatically. The third attempt puts into effect the concept of *Self-Attention* in GAN [92], but with a modification, since a *Cross-Attention* is obtained in order to give more importance to z^l over x^l, $x^l_{z^l} = CrossAttn(x^l, z^l)$ and additionally more importance of x^l over z^l, $z^l_{x^l} = CrossAttn(z^l, x^l)$. The concept of cross-attention will be preserved in every developed model and in this case, the Cross-Attention GAN (CAGAN) is built. It is applied on the previous architectures to further enhance the generative and approximation capabilities. Many other architectures have been developed, but they are not satisfactory enough to be reported.

4.2.2 Proposed approach: SiamCAGAN

The GAN based approach itself is not sufficient for a good estimation of target localization through mask generation. Starting from the idea of Siamese trackers, a single encoder branch has been developed to encode search and target images preserving the notion of cross attention before the convolutional operation in each layer, that is used to halve the size of the inputs. The single encoder should process the two images in similar way, also taking into account the cross influence of each one over the other. Unfortunately, results are not as promising as expected. The last attempt is to use the ResNet50 as backbone and the resulting features map will be cross-correlated by

the encoder. To give stronger feedback in the learning phase this network has two output branches, one for the mask generation and another for background/foreground classification. In this case, the intermediate layer outputs of the resnet in mask branch are used to be concatenated in the decoder to reinforce the reconstruction.

4.3 Proposed approach: SiamCA

Generative approaches are not enough if used as main component. Following the state of the art the research path moved to more flexible solutions based on Siamese architecture with dilated kernel pretrained on ImageNet as backbone. In this approach the purpose is to preserve the Cross-Attention module and use it before the correlation. Most of the trackers rely on a Region Proposal Network (RPN) to regress the bounding box. In the proposed strategy, the aim is to have an anchor-free tracker, which has three output branches with mask generation, bounding box regression and FG/BG classification. Mask generation is a group of upsampling interpolations and convolutional layers that at each step are added with the resnet output for the corresponding spatial dimension. Bounding box and classification are two convolutional neural networks to estimate a single bounding box and a class score map of reduced size. It is believed that branches influence each other even if there is not a direct link among them. This makes it possible to tune a single hyperparameter and get an improvement of the others in a predictable positive-negative way.

4.4 Proposed approach: Temporal relation

Classification and regression tasks could be considered to be directly correlated. The output classification score map is used in the tracking phase too so as to work in combination with the regression and to choose the best bounding box based on the best probability score. The correlation is obtained by extrapolating the *global localization context vector* of the cross-correlation output of target over image. In parallel, a

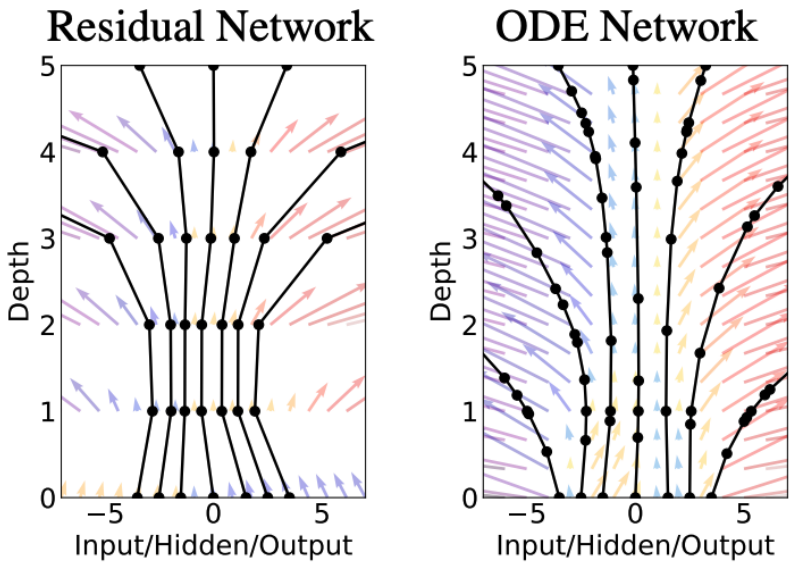


Fig. 4.2 Comparison of vector field representation in discrete residual networks and odenet [11]

recurrent neural network with long-short term memory capabilities (LSTM) is used. In the end the output of the LSTM is added to the context vector and the result is included in a classification branch and in a regression branch. The aforementioned global vector is obtained by utilizing a *Global Average Pooling* as stated in [94]. This is due to the fact that it is able to better preserve localization information thanks to the average operation, which does not hard-cut the network response. The idea is to strengthen the recurrent features with localization patterns and use the memory information to extrapolate the relevant information for the two different tasks.

4.5 Ordinary Differential Equation Networks

ODENet [11] exploits how dynamical systems can be of great interest in training neural networks. They remove the necessity of specifying a discrete number of hidden layers. Neural networks are used to parametrize derivative of hidden states. Residual networks sequentially perform operation on the hidden state $h_{t+1} = h_t + f(h_t, \theta_t)$ where $t \in 0, ..., T$ and $h_t \in \mathbb{R}^D$, which can be seen as Euler discretization of continuous operations [52]. Adding more layers and increasing the model complexity is tantamount

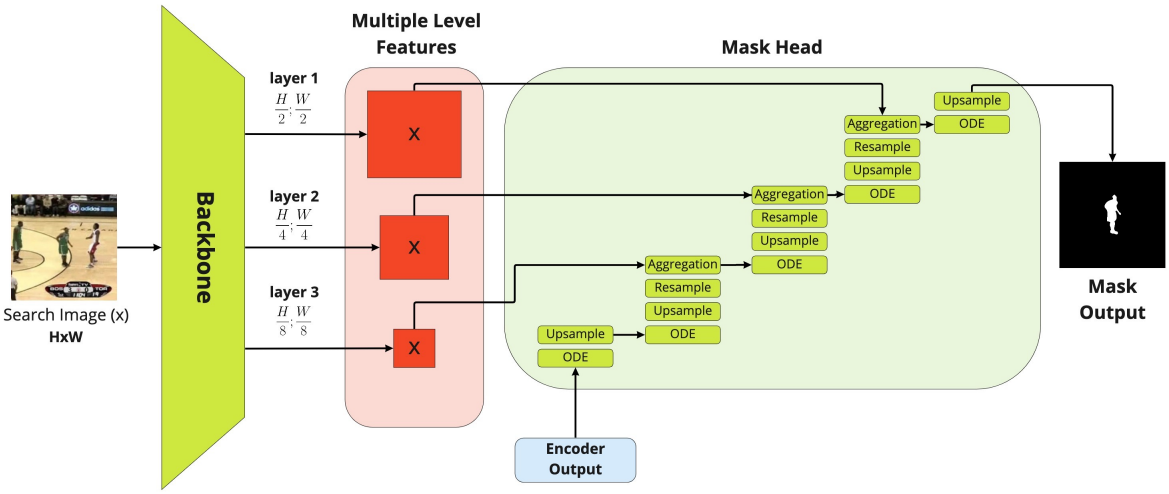


Fig. 4.3 Generic Mask Head with ODE module

to parametrizing the continuous dynamics of hidden units by an ordinal differential equation

$$\frac{d\mathrm{h}(t)}{dt} = f(\mathrm{h}(t), t, \theta) \tag{4.1}$$

where $t = 0$ is the input layer and $t = T$ the output layer in a neural network on the condition that $\mathrm{h}(t)$ represents the ODE solution at some time T. The final state can be computed by an ode solver, evaluating the hidden units dynamics f for the determination of the solution, as shown in figure 4.2.

4.5.1 Proposed approach: ODE Mask Generation

In literature [60, 61], many studies have demonstrated the benefits of relying on low-level features aggregation to generate fine mask reconstruction. It allows to incorporate the spatial information in low-level features that represent high-level semantic information. This aggregation could also be seen as a reconstruction of residual information [88].

Along this line of reasoning, an ode module is studied to make it possible to embed a continuous space residual reconstruction to get a fine-significant output mask. In the proposed case, *rk4* integration method is used with no adjoint learning schema due to the instability of the method in similar contexts [73].

Based on a simple reconstruction architecture as explained in section 2.4, the main contribution is to modify the residual part of the algorithm, following the schema in [24] and introducing a composition of convolutional layers, batch normalization, and some kind of attention to be evaluated with an ode solver as a time dependent function, with $t = 1$.

4.6 Proposed approach: Geometric Constraints

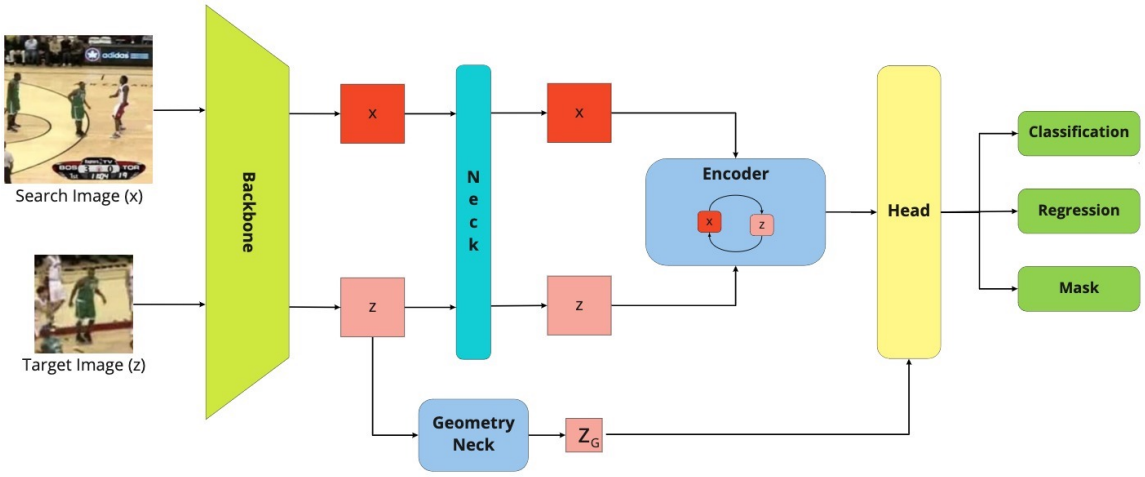
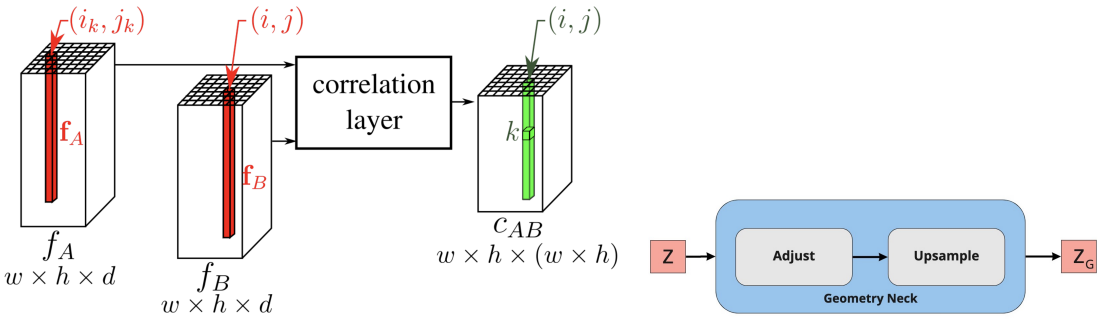


Fig. 4.4 Siamese structure with geometry neck add-on

The possibility of adding geometric constraint is explored. During the tracking phase, no constraints or elements that help to learn modification through time are usually implemented. It is left to the tracking phase the burden of the visual model update, introducing penalties based on score maps or online learning. This is based on the idea in [69] where it is used for geometry matching and descriptor similarities, along with their spatial locations, are considered for geometry estimation. It is composed of a *correlation layer* followed by a normalization. As shown in figure 4.5a there are $f_A, f_B \in \mathbb{R}^{h \times w \times d}$ that are the resulting feature maps at the end of the encoder and the correlation C_{AB} can be seen as the dot product of the two matrices $C_{AB}(i, j, k) = f_B(i, j)^T f_A(i_k, j_k)$, where each component of one matrix is correlated with all the components of the other matrix. It is modified to add a self-attention



(a) Geometry constraint [69] (b) Geometry Neck

layer on the output based on Efficient Channel Attention (ECA) [64]. Usually, it is used on convolutional layers, but it gives an efficient local cross-channel interaction in order to better extrapolate stronger relations on geometric correlated features. To give stronger feedback, the geometric constraint is inserted at the end of the encoder after the correlation is performed.

In the architecture the two inputs are represented by the encoder output and modified target features. Those features are the result of a new component in the network, which is called *Geometry Neck*. It is placed at the same level as the common neck (figure 4.4), but it downsamples the target on features dimension, but upsamples it on the spatial dimension in order to match the size of the encoder output and make the correlation possible (figure 4.5b).

The geometry constraint layer is tested in various solutions. In some architecture it acts differently in the training phase, where the input target is used. The same happens in the tracking phase, where the target with the alteration is used and an online learning is expressed by an alternative module, as it is explained in section 4.7.2.

4.7 Online learning

Online learning seems to be an important part of the tracking process. In a video sequence the target undergoes many modifications due to its movement, the movements of other objects, movements of the camera and other type of alterations. Updating the visual model by learning the target alternations can lead to reduce the problems that have been discussed in section 1.1.1. Consequently, it allows to minimize the chances of losing the target.

4.7.1 Learning Target Dynamics

Keeping track of alterations and try to correct them is not applicable, but it is possible to use a fast general transformation learning model that allows target variations learning and background suppression, completely shifting the focus completely on the target from previous frames [28]. It extends the common matching problem to a dynamic Siamese matching process:

$$S_t^l = \text{corr}(V_{t-1}^l * f^l(O_1), W_{t-1}^l * f^l(Z_t)) \tag{4.2}$$

where S_t^l is the response map, V_{t-1}^l is the *Target Variation Transformation* and W_{t-1}^l the *Background Suppression Transformation*. Both update the visual model learned from original image O_1 and search region Z_t. This modification stimulate $f^l(O_1)$ to be similar to $f^l(O_{t-1})$. Those transformation are learned using *Regularized Linear Regression* [74] by a fast circular convolution operation in the frequency domain. In figure 4.6, it is shown the effect of the transformation on the target image O_1 to O_{t-1} and the search region Z_t with a cropping centered on target at previous frame G_{t-1} and a Gaussian weight map applied to highlight the foreground object $\bar{G}_{t-1}$.

The online module is attached to various developed trackers to check the contribution of learning target transformations during time. A modification is also implemented adding a memory parameter to keep track of multiple variations and using multiple targets at different times. The first element is always assigned to O_1 in order to preserve

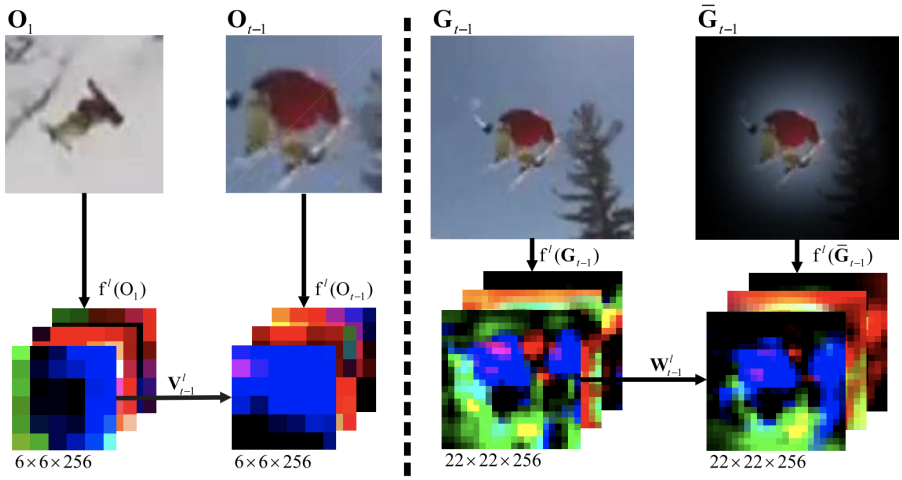


Fig. 4.6 Target Appearance Variation Transformation and Background Suppression Transformation [28]

the capacity of the original method, having additional k elements. Each element is shifted to the previous position to keep the memory updated. The best result is chosen by taking the variation with max response after the circular convolution so as to be used to update V_{t-1} variation transformation.

4.7.2 Proposed approach: Learning Target Geometry

The learning of target variations is modelled using *Contractive Autoencoder* [68]. They help to learn a compact representation of the data that can be used for dimensionality reduction and to find a latent space where similar data have near vector representations. Opting for the contractive version means to choose a penalty term that corresponds to the Frobenius norm of the Jacobian matrix of the encoder activations with respect to the input. The penalty helps to find a representation that best expresses data local directions variation, corresponding to a lower-dimensional non-linear manifold, while being more invariant to the vast majority of orthogonal directions to the manifold. The autoencoder learns various augmentations of the initial target to best represent it if some alterations arise in the next frames. After the initial learning phase, geometry neck weights (section 4.6) are used to build the encoder part of the autoencoder. It

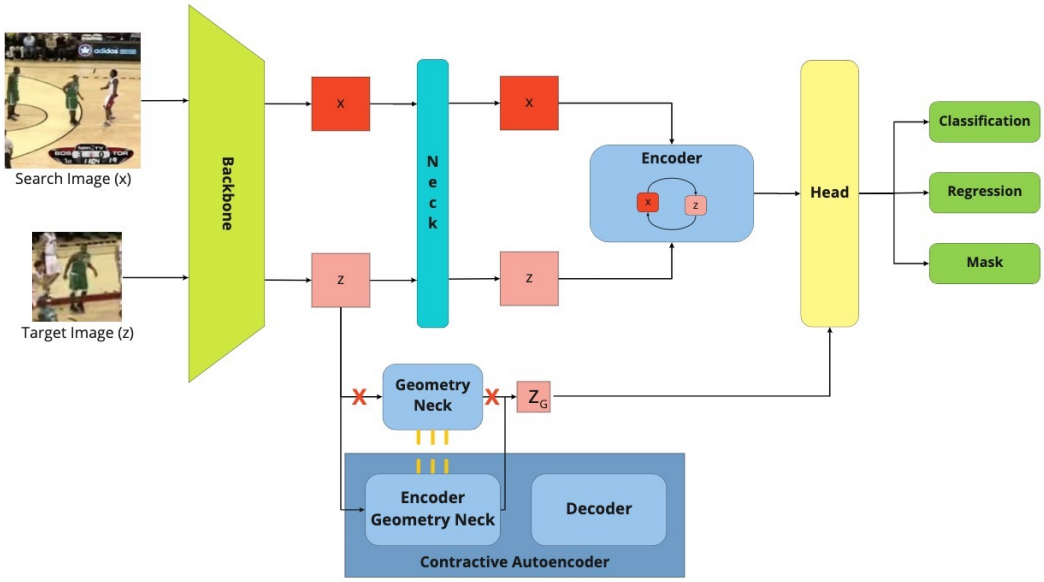
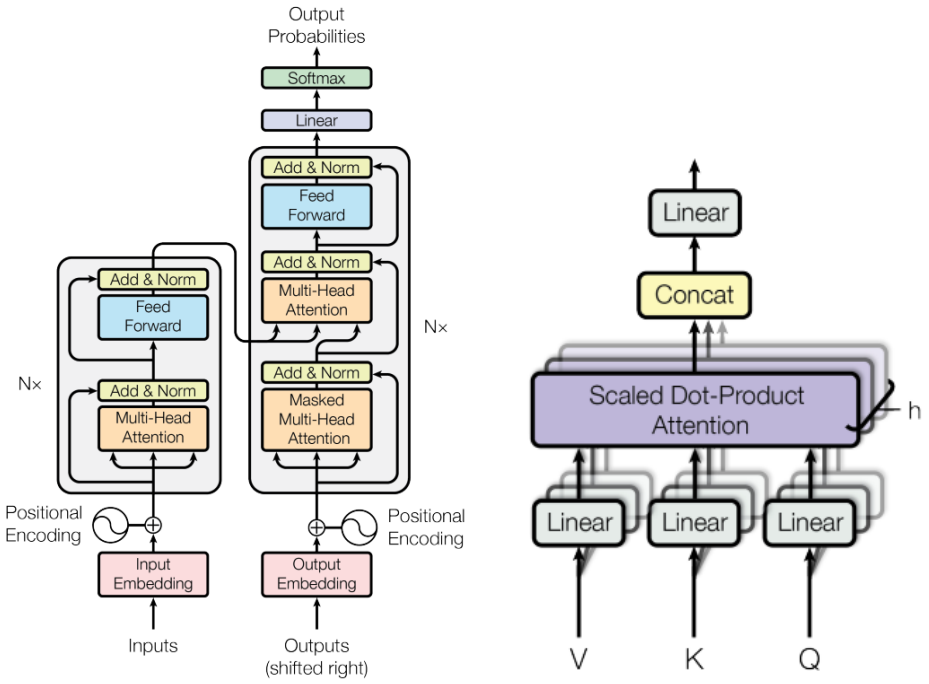


Fig. 4.7 Modification of tracking pipeline with online target geometry learning

ensures that geometry neck is fine-tuned on various target representations. Following this strategy target features are processed by the new tuned version of the geometry neck in order to get a compact representation generated by the encoder subspace. This is injected into the tracker body as input for the geometry constraint layer (figure 4.7). At each step, the autoencoder is fine-tuned with the next localized bounding box, which is yet augmented to provide many random different modifications.



(a) Transformer structure from [79] (b) Multi-Head Attention from [79]

4.8 Transformers Networks

Transformers [79] are widely used in Natural Language Processing. This architecture has completely changed the capacity of a neural network to work with text values. In the original form, it consists of an encoder part and a decoder part. In both there are many internal layers that are composed of a well studied structure (figure 4.8a) with a positional embedding on the input, because the internal operation is not able to recognize and remember the position of the input values. In each layer, there is the so called Multi-Head (Self) Attention, which is a composition of matrix products where the correlation of the data is estimated by itself. After that, the normalization of the residual is computed and passed to a simple Feed Forward Network. In the end, the normalization of the residual is computed again.

The decoder structure is similar to the encoder, excepts for the initial Attention that can be masked to estimate the masked value. The Multi-Head Attention module is a *cross attention* of the encoded representation of the input values and the decoding elements on which they should be correlated. In this model, the attention plays the

most important role because it is the key to the success of this architecture. Nowadays, most studies are conducted on how to enhance the attention calculation of finding numerically stable alternative to it. Attention is an easy and effective concept that has been used for years [14, 84, 87], but it reached the maximum of its power within transformers architecture. In figure 4.8b, the structure of the attention is explained. Inputs are processed by three linear projections to get the right number of features that must be decomposed by the number of the heads. The heads are the components that help to give a different correlation index for each value. To enhance the final result, it is as the problem is broken down into smaller ones, allowing each one to be able to extract a proper correlation. Transformers solve the long-term memory problem of recurrent networks. The structure is able to propagate the temporal information through the many layers of which it is composed of, reducing the risk of forgetting information.

Over last years the research community has started to use this architecture in computer vision tasks as well, obtaining excellent results. In the next section, there will be description of many methodologies applied in the tracking field.

4.8.1 Vision Transformer

Vision transformer (ViT) [20] is an attempt to put into effect the power of transformers to work on images. The aim is to classify images using only an encoder. The structure is shown in figure 4.9

One of the interesting components is the position embedding, that in the case of an image becomes a *patch embedding* where the image is divided in patches of a specific size. A flattening and a final projection are applied to bring the number of channels to the desired number of embedded features. All patches are used in the encoder along with an additional token that learns the classification representation of the data. In the end only the classification token is retained and passed to a *mlp head* whose result is the image classification.

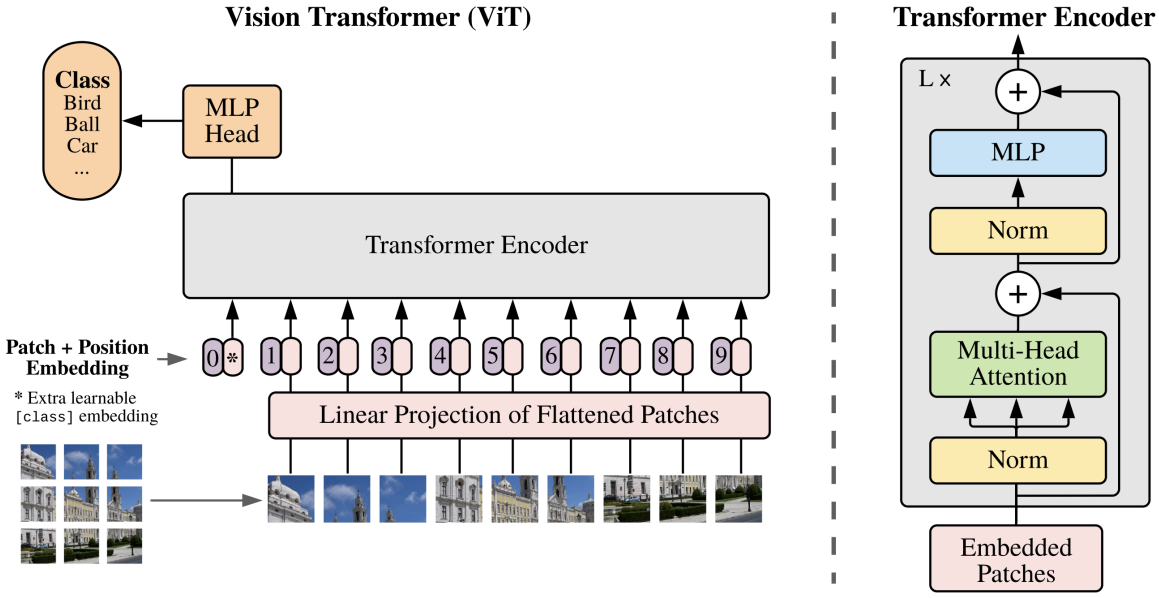


Fig. 4.9 Vision Transformer [20]

In the current research many architectures have been studied to be suitable for tracking and multi-branch outputs. The first step that is common to all models is to substitute the patch embedding of a plain image with patch embedding of the extracted and adjusted features originated from the backbone and neck sides.

4.8.2 Proposed approach: MCRViT

One of the limitations of ViT is that the output token loses all spatial information, since it has the individual classification information. In the case of tracking, however, this is a problem, as this information is still needed in order to get the output of the various heads. To solve this issue, it was necessary to move the compression of the spatial features within the encoded feature space. Unfortunately, this leads to the explosion of the number of features to process, reducing the overall value of the hyper parameters available, such as the internal features number and the internal layers number.

The main idea is to enrich the base architecture by adding new additional tokens that are specialized for tasks of each output branch in order to obtain learned parameters.

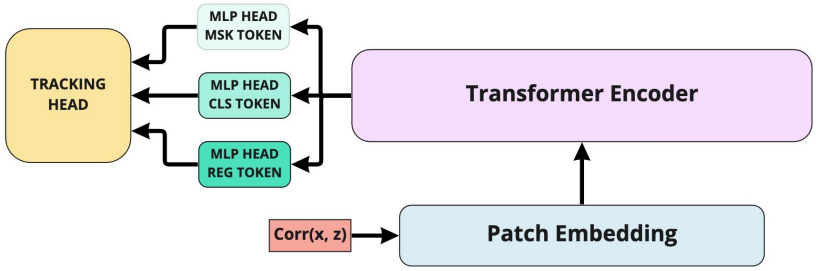


Fig. 4.10 MCRViT Structure

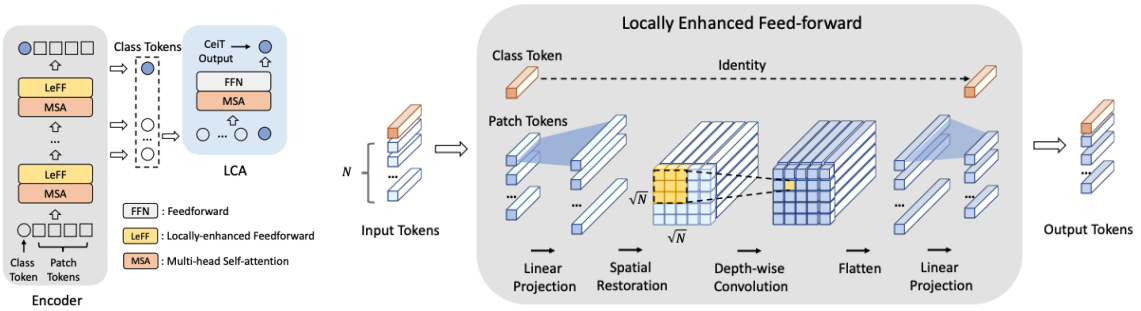
There are from two to three tokens one for classification branch (similar to the original model) one for regression branch, and another for mask branch. The MLP head can be removed based on the network design. The architecture is shown in Figure 4.10 and is called *MCRViT*.

4.8.3 Proposed approach: Shared Vision Transformer

As stated before ViT demonstrated great performance on image classification, taking the advantage of learning a component specialized for classification which uses an encoder-only structure. In addition, it could be utilized to extrapolate more discriminative and important features with the possibility of correlating two different elements that are ideally linked together. The encoder encapsulates the relation between search image and target features with an iterative correlation by encoding them with the same computing layer, especially after taking advantage of multi head cross correlation ability to correlate target over image continually. In this situation, the class token and the mlp head have been removed.

4.8.4 Convolution-enhanced image Transformer (CeiT)

As in this case with many promising tools, ViT is continually studied and enhanced by the research community. One of the most innovative researches is called *Convolution-enhanced image Transformer* (CeiT) [91]. It involves convolution to extract low-level features, empowering the locality with the ability of Transformers to model long-range



(a) Layer-wise Class token Attention [91]

(b) Locally-enhanced Feed-Forward layer [91]

dependencies. It provides three fundamental modules (the whole architecture is in figure 4.11a):

- **Image-to-Tokens** (I2T) module: it extracts patches from low level features
- **Locally-enhanced Feed-Forward** (LeFF) layer: it promotes the correlation among neighboring tokens in the spatial dimension. It is substituted with FFN in ViT reshaping the feature vector in a matrix that is treated as an image and a Depthwise convolution is applied. In the end, a new flattening is performed (figure 4.11b)
- **Layer-wise Class token Attention** (LCA): it uses the multi-level representation of the attention at the top of the model. It is based on the concept that each convolutional layer depends on the previous one with a influence inherited by the receptive field size. Taking into account the spatial relation in convolution and relations that are embedded in each attention module, it relies on the full history of token attention to strengthen the final token, avoiding the possible loss of important information in the dependency path built by the transformer

4.8.5 Proposed approach: Convolution-enhanced image Cross Transformer (CeiXT)

Starting from the CeiT structure, the proposed embedding in a tracking architecture is to substitute the I2T module with the feature extracted from ResNet50 at *layer3* in order to not enforce high-level features, eliminating the necessity of CeiT. There are two distinct inner encoders to process the search features x and the target features z independently. After the encoding phase, there is a modification of the standard encoder with cross correlation operation. By using multi head cross attention with image and target, the LeFF module is modified in a *Locally-enchanced Feed Forward* ***Cross Correlation*** *Network* (LeFFX) where the target z is used as weight matrix for the depthwise convolution, as it usually happens in standard, cross correlation module. It enforces the features represented by the cross attention of the target z over the search x_{z_attn}. Furthermore, three tokens have been used as in section 4.8.1 for mask, class, and regression; three LCA modules are applied on each token, taking into consideration the attention history of each token. This architecture is called *MCRCeiXT*.

4.8.6 Proposed approach: Shared Cross Transformer

The shared cross transformer splits each layer in two submodules: the first follows the classic transformer; the last is a *Cross Encoding Layer* that uses cross attention of x over z and cross attention of z over x. Iteratively, each transformer layer computes the separate encoding and the aggregation of features. At the top of the encoder, there is the last cross encoding layer and just the search image is preserved in the end. The same layers process both search image x and target image z. Two sinusoidal embeddings are employed for each input and the decoder is not present in the architecture.

4.8.7 Proposed approach: ViTCRT

As it has been shown in paragraph 4.8.3 Vision Transformer is deployed as a base to model the encoder part of the tracker in order to be able to correlate both search image

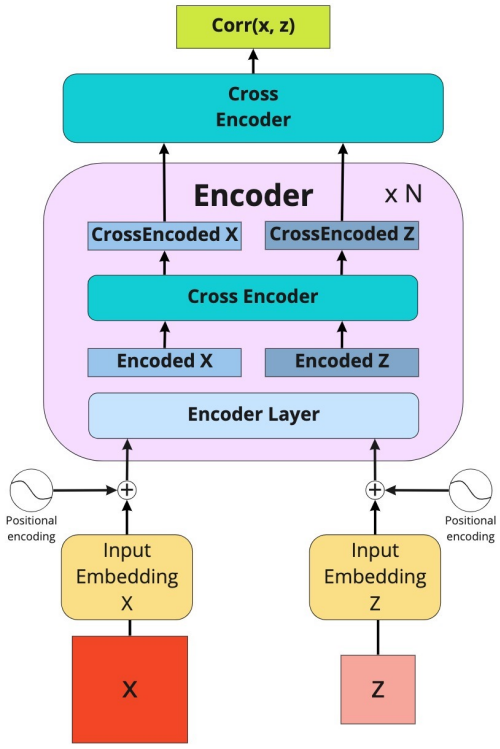


Fig. 4.12 Shared Cross Transformer architecture

feature and target image feature. Two tokens have been used, one for classification and another for regression. Based on the strategy in [85], the patch embedding is reduced to have size 1; it is further computed on backbone output feature maps, but this is equivalent to taking a patch of one pixel size on an image. The next step is to concatenate on length dimension, both features in a unique vector that is used as input for the transformer. It ensures that features will be *naturally* correlated by using multi-head self attention, which in our case, become a mix of self-attention and cross attention. It computes simultaneously the image self attention along with target and cross attention of target over image and image over target. In this specific case, it helps to compute the importance of the classification and regression tokens over both pieces of data, where each element expresses its relevance towards the desired. In figure 4.13, the architecture is shown. It is extremely relevant because it directly influences the tokens. At the top of the encoder tokens are extracted with the learned representation of the pixels in the transformer. One important difference with MCRViT

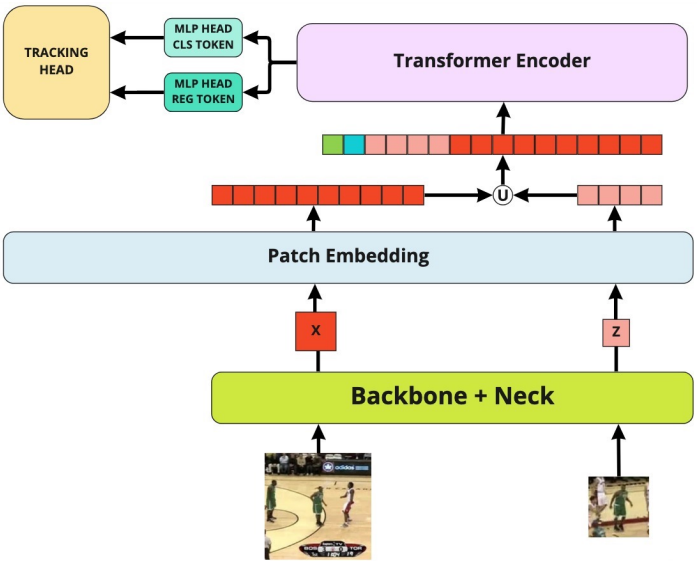


Fig. 4.13 ViTCRT Architecture

is the embedding space, because this new version relieves the burden of solving the spatial incompatibility of the token with the head. Differently from the original ViT, which employs the last MLP head as output layer for classification, the tracker needs to further process tokens on spatial dimension too so as to create appropriate feature maps that are a sub-space representation of the initial search image, where encoded information on foreground/background scores and bounding box location based on the technique in [86] are present.

Chapter 5

Models Architectures

> If you're not trying to be real, you don't have to get it right. That's art.
>
> *Andy Warhol*

All methodologies reported from the previous chapter are part of multiple architectures, in order to understand what works and what does not in combination with the visual tracking task. For each experiment a figure and an explanation of the concept is reported.

5.1 Architectures

The following explanation describes how the various proposed methodologies reported in chapter 4 have been arranged to build multiple architectures, which are investigated to understand how they can contribute to the tracking area. All architectures put ResNet50 backbone into effect with or without convolutional dilation. Layers of ResNet50 are fine-tuned on the tracking task. There are three main structures:

1. **SiamCA**. It uses a Siamese Cross Attention encoder before the depthwise cross-correlation in all experiments.

	Legenda
SPL	Simple Encoder
CA	Cross Attention Encoder
ODE	ODE in Mask generation
SUM	Features sum in Mask generation
CAT	Features concatenation in Mask generation
RCT	Recurrent LSTM layer
ATTN	Efficient Channel Attention in Mask generation
ODYN	Online Dynamics Learning (**M**: Multiple)
OAE	Online Geometry Learning
MTL	Multiple Layers

Table 5.1 Experiments Acronyms Caption

2. **SiamSPL**. It does not use cross attention before the depthwise cross-correlation. It is used as an *ablation study* to check whether or not the cross-attention module is important in developed architectures.

3. **Transformer Correlation**: The correlation is performed through a transformer before the head module.

5.1.1 SiamCA-Base

SiamCA is a Siamese Cross Attention tracker. It uses a ResNet50 backbone with dilated convolution. The encoder is a simple cross attention module with the last cross correlation operation. It has a basic three-branch head output. It relies on a series of Convolution-BatchNorm-ReLU operations to generate the output bounding box regression and the output classification map. The methodology is expressed in the state of the art for mask generation.

5.1.2 SiamCA-Base-DYN

SiamCA-Base-DYN is equal to its counterpart in section 5.1.1 on offline part. The difference lies in online tracking where the learning of target dynamics (section 4.7.1) has been added (as shown in fig. 5.1).

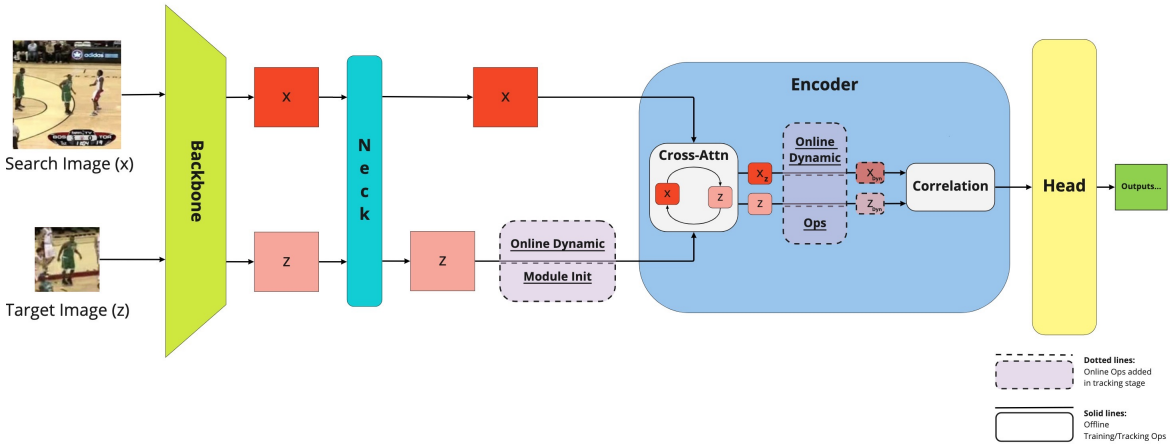


Fig. 5.1 Integration of Online Target Dynamics learning in the network architecture. *Dotted lines* represent modules which are activated only in tracking stage

Multiple-Dynamics

Multiple-Dynamics (ODYNM) is referred to the attempt to add multiple temporal targets with the aim to learn the best transition from the state T_0 to T_{n-l} where n is the previous target and l is the state of the target in a moment before n is stored in a memory buffer.

5.1.3 SiamCA-ODE

Mask branch of SiamCA is enriched with ODE layers (sec. 4.5). In this implementation the sum of low-level features and high-level backbone features is preserved.

SiamCA-ODE-SUM

This sum is enriched with an ode function composed of a single convolutional layer with a kernel size of 1 in order to resample the features by continuous operations.

- **SiamCA-ODE-SUM-DYN**. Dynamic module is added in online tracking to check the compatibility and the contribution that it can give to a more continuous approach.

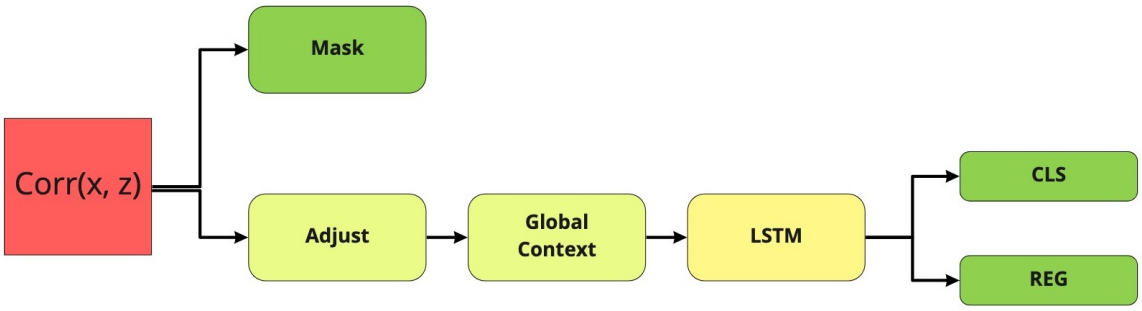


Fig. 5.2 Recurrent Head

- **SiamSPL-ODE-SUM**. An ablation study is performed removing the cross correlation attention module and using the simple encoder

SiamCA-ODE-CAT

High-level features from backbone are concatenated with the mask reconstruction as in skip-connections. After that they are processed with an ODE function consisting of a convolutional layer and a batch normalization. This occurs at any level before being upsampled.

SiamCA-ODE-CAT-ATTN

In the case of mask generation, the aim is to start with a set of features that are globally representative and spatially consistent. For this reason, the global context is computed by an average pooling and it is later used as a starting point for the computation. The ODE function is modified with the following operations Conv-BatchNorm-ECA. The Efficient Channel Attention is a cross channel attention mechanism that is implemented by 1d convolution. Concatenation is employed for skip-connection with backbone high-level features.

SiamCA-ODE-SUM-ATTN-RCT

The only difference in mask is the use of SUM operator instead of concatenation. The main changes are on classification and regression branches. To enrich the correlation

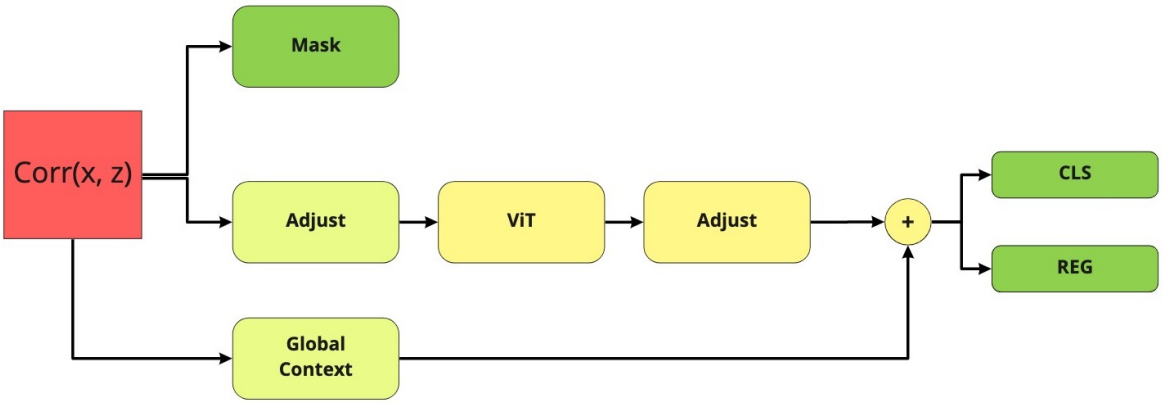


Fig. 5.3 SiamCA-ODE-SUM-ATTN-ViT

of both and try to put an element that helps to memorize the recurrent pattern in the localization and classification, a recurrent layer (consisting of an LSTM layer) is implemented. The head also tries not to lose important spatial information through the flattening and reconstruction of the signal, thanks to the LSTM, which adds the global context to the encoder output. The architecture is shown in figure 5.2.

- **DYN** version is also tested

5.1.4 SiamCA-Transfomers

Transformer networks has been used to maintain a strong correlation based on a compact representation and to remember recurrent patterns. In addition, in this section many architectures have been tested and the most relevant have been reported.

SiamCA-ODE-SUM-ATTN-ViT

Mask branch has the same structure shown in paragraph 5.1.3. A vision transformer is used, following the classic structure with one output token (figure 5.3). This token is spatially enriched, as demonstrated in previous paragraph, by using the global context. At the top of the head the token has been given input to classification and mask branches.

SiamCA-ODE-SUM-ATTN-MCRViT

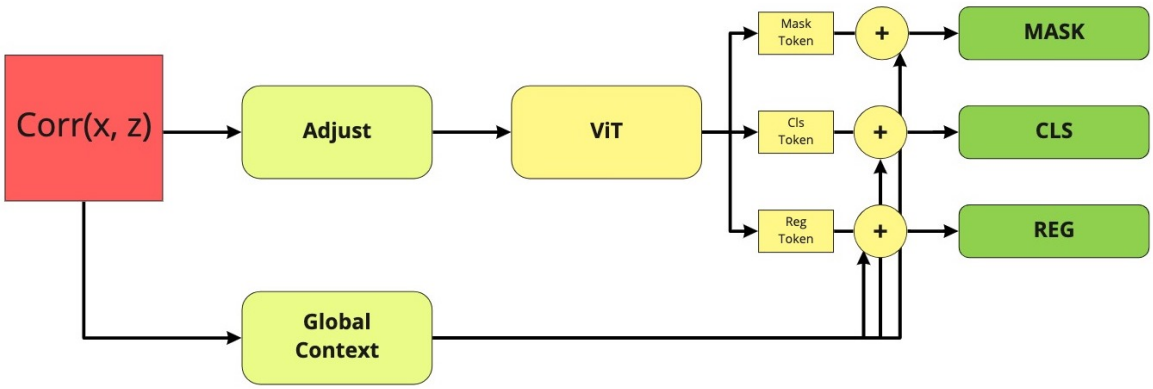


Fig. 5.4 SiamCA-X-ATTN-MCRViT

The whole structure, including the reason for inserting global context, is identical to 4.8.7. The difference is that multiple tokens have been learned by Vision Transformer. Instead of using just the *classification token* as a "global" token, there are three particular elements that will be learned for a specific task: one for mask, one for classification and one for regression.

SiamCA-SUM-ATTN-MCRViT

It is conducted as an ablation study on ODE mask generation. The structure remains the same as in paragraph 5.1.4, but mask ode function is removed from the branch. The efficient channel attention is moved after the layer upsample in order to make output upsampling convolution stronger on channel dimension. In figure 5.4 the architecture is shown.

- **SiamCA-SUM-ATTN-MCRViT-MTL**. Same structure of the above experiment, but a greater number of layers is employed in transformer, preserving all other components and hyperparameters. It acts like a comparison test with the one layer version.

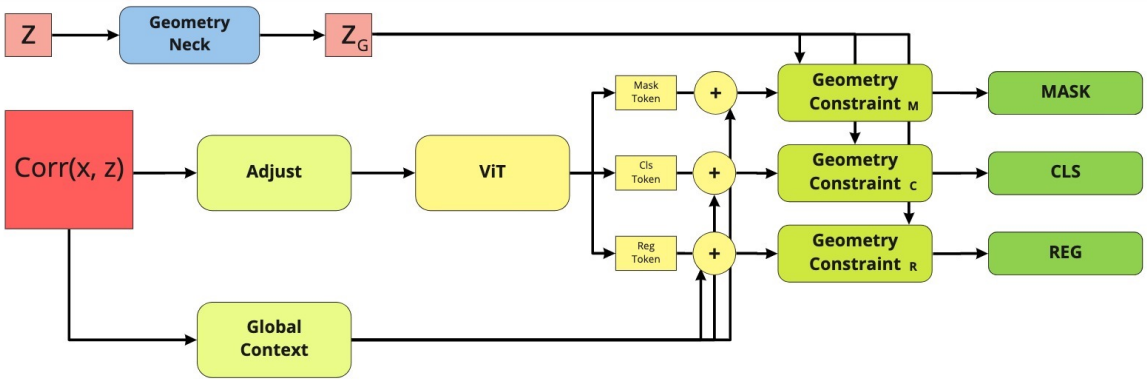


Fig. 5.5 SiamSPL-VTGC

- **SiamSPL-SUM-ATTN-MCRViT**. As an ablation study the cross attention encoder is removed and the simple version is implemented.

5.1.5 SiamSPL-VTGC

SiamVTGC is the acronym for *Siamese Vision Transformer with Geometry Constraint* (figure 5.5). The structure follows the other transformer implementation, using MCRViT with the global context for spatial consistency. Before head computation, tokens will be geometrically constrained (sec. 4.6) through the target features refined by a geometry neck. This can be considered a kind of attention over the target structure.

- **SiamSPL-VTGC-OAE**. The online version uses an autoencoder to learn the target variations and to to substitute the geometry neck with an updated invariant version of the target features. An additional use involves geometry constraint
- **SiamSPL-VTGC-MTL**. A multilayer version of SiamVTCG
- **SiamSPL-VTGC-MTL-OAE**. A multilayer version of SiamVTCG with the online learning module

5.1.6 SiamSPL-GC-FCOS

It does not use transformers for middle computation. It preserves the Geometry Constraint schema (Geometry Neck + Geometry Constraint). Mask is computed after

the constraint, whereas classification and regression are done using FCOS (sec. 2.3.3) detector.

5.1.7 SiamSPL-VTGC-FCOS

The architectures is the same as 5.1.5, with the difference that the FCOS detector is used as head and the inputs are the classification and regression tokens.

5.1.8 SiamConnector-MCRCeiXt

The encoder is a modified transformer named MCRCeiXt (sec. 4.8.5). To summarize the transformer has two parallel encoders that take care of search image features and target features, correlating them using cross attention. The output uses FCOS as detector.

5.1.9 SiamSharedTransformer

One of the most simple structures, it uses a shared transformer to encode both inputs and to cross-correlate them at each step. The output is a single element that is deployed by FCOS. No mask generation in this architecture.

5.1.10 ViTCRT

It shares similarities with MCRViT as it uses the same structure, but the encoding is more based on a transformer baseline approach where spatial dimension is flattened and encoding is all on embedding dimension. Extracted tokens undergo a multiplicative spatial attention thanks to the encoded features performed by ViTCRT . It reconstructs the spatial dependencies of the tokens. A structure similar to Alpha-Refine (section 2.3.4) is used for bounding box regression. Classification is a simple MLP, outputting a vector that will be reshaped to fit the image size used as score map. It has been built with 6 layers in the encoding and 8 heads. To search the best architecture, many of

Tracker Config	MLP	HEAD ATTN	CLS	AUC	P_{norm}	OP50	OP75
baseline	Full	Spatial	MLP	53.91	64.36	66.0	39.4
box_mix	Full	Spatial	BOXMIX	51.12	48.64	49.33	25.63
multihead	MT	Spatial	MLP	46.48	56.11	57.6	33.17
tokens	Tokens	Spatial	MLP	56.05	67.54	68.75	40.75
nomlp	none	Spatial	MLP	57.06	68.65	69.83	40.47
nomlp token attn	none	Token	MLP	59.15	71.70	72.37	41.88
nomlp masking	none	Spatial	MLP	56.73	67.47	69.11	40.26
multihead token attn	MT	Token	MLP	**67.34**	**81.71**	**84.17**	**52.38**

Table 5.2 Comparison of ViTCRT configuration on OTB100. **MLP** is the final mlp head in ViT possible values are *Full* to process the full sequence, *Tokens* use one head to learn a representation for all tokens, *MT* creates a different mlp head for each token. **HEAD ATTN** is an experiment on spatial reconstruction of token after ViT output, *Spatial* means the encoded features multiplied by the attention, *Token* use the token representation to reconstruct the matrix. **CLS** represents the use of class prediction using a single *MLP* or *BOXMIX* weighting the output MLP with the probability scores from regression head.

them have been tested on OTB100 in order to have fast and relevant results (table 5.2):

- With / without mlp heads in ViT
- With multiple mlp heads in ViT
- Weighting the class score map using the bounding box scores
- With / without attention masking

The last configuration, in table 5.2, seems to be the most relevant. More specifically than mentioned above, it is composed of a multiple head in transformer where each token can learn a proper representation without being mixed with others. This should enforce the task-capability of each one. The ViT outputs are two tokens without length; they just have an embedding representation, which is problematic because heads need to process spatial features as well. For this reason, the feature spatial representation is also returned and multiplied with tokens to get a lightweight attention. Then, it is multiplied again with the token, ensuring that the token has a spatial dimension too. In an experiment (box_mix), a different type of head is used in training, generating

a feedback from the regression head where the probability score maps are used to weight the output of classification. The final scores are summed with a weight of 0.5. Unfortunately, this has not proved to be the best solution.

5.2 Training

For the training stage an offline approach has been chosen, like in many of the most important techniques. It consists in showing single images to the network with the corresponding target. The target representation depends on the type of dataset and for those that are based on single image the exact matching target is used. Conversely, for those based on video sequences, target is not the exact image that is present in the search image, but a sampling. The strategy is used to choose the same object in a different position over time. The main strategy is called *causal* and consists taking a frame at time t_i as reference image and choosing the next valid frame index t_i^v as a starting point for sampling the target image. A "valid" frame means, one where the target is present because this is not true for all frames in a sequence, in fact, some of the datasets are both for short-term and long-term tracking or just for object detection or segmentation. Starting from the first valid index the next frame at time t_{i+n}^v is chosen. This strategy guarantees both to teach to the network how to precisely identify an object based on shape and position. It further helps to reduce bias in the localization based on the exact shape of the target, forcing the network at the same time to find the best candidate with the most relevant and similar shape in the frame. For training, a collection of different datasets has been used jointly to allow the network to see as many different samples and objects as possible. Moreover, the sampling strategy diversifies what the true sample and the related target are. In addition, data augmentation artificially enriches the dataset and, most importantly, it reduces the bias, since all images have a hypothetically similar context. The image is randomly jittered in any direction and to a different scale, adding black border to fit the expected size of the image. The same process can be applied on target. It also avoids that the

target is always at the center of the image, reducing another possible bias introduced by the data. From the dataset, bounding box annotations and mask information are extracted with the same pre-processing operations in a way that is applicable to the two types of data in order to make all input elements consistent with each other. After data have been computed by the neural network the output is learned with appropriate loss functions or multiple losses, for each branch. For most of the presented models the loss does not change. For all models, except for ViTCRT , Stochastic Gradient Descent with Momentum [63] was used as the optimizer. For ViTCRT model, ADAMW [51] was used with 500 epochs at a learning rate of 1e-4 with a step learning rate decay of 0.1 every 200 epochs.

Computational Costs

One of the biggest problems in tracking is the computational resources required to perform the training phase. It must be taken into account that a tracker is a collection of many components, some of them already quite large on their own, such as the backbone. In addition, the datasets used are large, leading to a fairly high number of samples per epoch. This means that being able to perform a single experiment is not at all simple. Most of the experiments carried out took hours and hours for a single epoch. This implies that even a small number of training steps required weeks and months of training time. Obviously, this involved a trade-off betweeen the time used to research new solutions and the time needed to test their effectiveness. In the results section, statistics at a different number of epochs are reported for some trackers. A low number of iterations are often found, which can lead to results that do not fully express the capabilities of the technique itself. For example when it was decided to work using transformers, based on their formulation, a complex obstacle was faced, so much so that the number of epochs in the first experiments is very limited, as well as the utilization of the number of hidden layers present within it. Apart from the experiment with *SiamSharedTransformer* and *ViTCRT* all other experiments have been conducted using a single layer and a very low number of input features.

On the other hand the experiments called Multilayers have used two hideen layers. SiamSharedTransformer instead has been tested with 2 and 5 layers while ViTCRT has the basic architecture of ViT with 6 layers in the encoder. As reported in the section of the proposed methodologies (section 4.8.1), the main problem with the first set of transformers is that when using ViT, the final output is a single token that loses the length and the spatial dimension. Therefore, it has been encapsulate it within the encoding itself, thus exploding the number of features to be processed and lowering the amount of layers.

5.2.1 Loss Functions

For each task, one or multiple loss functions have been used with an appropriate weighting factor to differently impact on the total loss and to reduce excessively steep slopes, avoiding getting an extremely small contribution of a loss in comparison with another. The loss will be formulated as:

$$\mathcal{L} = \lambda_1 \mathcal{L}_{reg} + \lambda_2 \mathcal{L}_{cls} + \lambda_3 \mathcal{L}_{mask} \tag{5.1}$$

where, obviously, each loss can be formed by one or more task specific losses. This forces the network to learn from multiple error functions. As a result, the whole network models a composed function, where each element is optimized.

Regression Branch

The regression branch has to predict the bounding box where the target is located. To measure the error of the localization the intersection over union is used as the main measure. Many IOU loss functions exist, and three of them used in the experiments are shown as follows:

- **IoU**: as stated before (section 3.4), the IoU measures the overlap of the pixels number in the resulting bounding box estimated boundaries and the ground truth box. It does not take care of the noise in the bounding box such as background

elements or occlusions because the important element is just the location of the object. It is usually employed as distance metric and generally the IoU measure equation 3.1 of the predicted bounding box b_t and the ground truth bounding box b_{gt} is

$$\mathcal{L}_{iou} = 1 - IoU \tag{5.2}$$

In this specific case, an optimized version of the loss is used in [89] as likelihood in a cross entropy function. It minimizes the negative log likelihood:

$$\mathcal{L}_{iou} = -\log(IoU(b_t, b_{gt})) \tag{5.3}$$

- **Generalized IoU** (GIoU) [67]: it is a more general metric that tries to reduce the gap in IoU where there is no information, in particular when a bounding box does not overlap the ground truth or if the bounding box cover a part of the image that is outside of the ground truth. Both pieces of information can be used to represent a detailed behavior of the proposed estimation by including theme in the learning schema. This is carried out by computing the smallest convex shape C enclosing b_t and b_{gt}. Furthermore, the ratio of area of C is computed, excluding b_t and b_{gt} and dividing the total area of C. In the end the ratio is subtracted from the IoU:

$$GIoU : IoU - \frac{|C \backslash (b_t \bigcup b_{gt})|}{|C|} \tag{5.4}$$

consequently:

$$\mathcal{L}_{giou} = 1 - GIoU \tag{5.5}$$

- **Smooth-L1** [25]: in regression tasks, it is common to find the optimization of a norm, such as *L1* or *L2*, which are the *Absolute Mean Error* (MAE) and the

Mean Squared Error (MSE). They can be seen as distance from one point to another in the Euclidean space. The advantages of taking them separately are that L2 has less oscillation when values are small, L1 has steady gradients for large values and helps to introduce sparsity. It is possible to get both benefits by relying on the smooth-L1, which uses the bounding box values formulated as:

$$\mathcal{L}_{smooth\text{-}l1} = \begin{cases} \dfrac{0.5(b_t - b_{gt})^2}{\beta} & \text{if } |b_t - b_{gt}| < \beta \\ |b_t - b_{gt}| - 0.5 * \beta & \text{otherwise} \end{cases} \tag{5.6}$$

It can be seen that when $\beta \to 0$ it converges to a L1 loss instead when $\beta \to +\infty$ it converges to a L2 loss.

Classification Branch

For classification, the evaluation is done on a map of responses that identifies two classes: foreground or background. For this reason the classification can be treated as a binary classification problem classifying each single element in the output matrix. Generally, the loss is computedby using the cross entropy loss, namely:

$$\mathcal{L}_{ce} = -\sum_i y_i \log \hat{y}_i \tag{5.7}$$

An alternative is to use the Focal Loss [48], which is very popular in object detection and can be used in environments where there is an imbalance dataset. This may be the case of tracking because the number of background elements class is supposed to be greater than the foreground one. It modifies the standard cross-entropy loss in order to balance the loss obtainedd for widely expressed examples. It adds a factor of $(1 - p_t)^\lambda$ with $\lambda \geq 0$ to cross entropy:

$$\mathcal{L}_{fce} = -\alpha(1 - \hat{y}_t)^\lambda \log \hat{y}_t \tag{5.8}$$

where α is a balancing factor.

Mask Branch

This branch tries to reconstruct the input image with a binary mask that represents the background and the foreground element. The concept is similar to the classification branch because each reconstructed output pixel can be seen as a foreground/background class. However, it tries to have a refined output in order to obtain a fine segmentation representation of the target in the frame, taking care of the scale, position and occlusions. The loss is managed by means of the L1 loss, which in reconstruction achieves impressive results and from a visual point of view the reconstruction is smoother and less blurred than the L2 loss. The L1 is simply the Mean Absolute Error described by the following equation:

$$\mathcal{L}_{l1} = \frac{1}{N}\sum_{i=1}^{N}|y_i - \hat{y}_i| \tag{5.9}$$

GAN

When training the Generative Adversarial Network the loss is relatively different. The MiniMax game loss is used where the Generator tries to minimize the following function, instead, the Discriminator tries to maximize it:

$$\mathcal{L}\min_{G}\max_{D}(D,G) = E_x[\log(D(x))]E_z[\log(1 - D(G(z)))] \tag{5.10}$$

where E_x is the expectation over discriminator probability $D(x)$ of real data x. E_z is the expectation of the Discriminator probability over the fake values generated by $G(z)$. It derives from the cross entropy, mixing the probability of real and fake data, in order to bring the fake data distribution close to the real data distribution. Following the literature [26], the generator loss is modified to avoid sticking the network at the initial stage of the training, minimizing the quantity $\log(1 - D(G(z)))$ that is mathematically equivalent. As experimented the generator can minimize more than one loss function [33]. The loss is structured using the first term plus the L1 loss so as to get a more structured reconstruction. Many test have been done trying to optimize

a loss function that will give a significant contribution. The best combination resulted in using minimax modified with focal loss, L1 + L2 loss combination.

5.3 Testing - Tracking

Tracking is performed with a different pipeline compared to the training phase as mentioned in section 1.2. In this specific case, the tracking will be presented with ViTCRT because, based on the type of estimation, there are several ways to post-process the output and make it consistent to represent a bounding box. During the initialization phase, where the first frame of the sequence is passed to the tracker together with the initial bounding box so that it can extrapolate the initial target image, preprocessing it and extracting the features. This is done in advance because, unless the systems are updated, it helps to speedup the performance in the tracking phase since the target remains the same for the whole sequence. Pre-processing and a forward step are performed using the same modalities shown in training. Firstly, the bounding box is regressed and the values are brought from the normalized interval $[0, 1]$ to the search image size. Regarding the classification branch the network has been tested with and without it. The update schema in STARK [85] is tested. Then, the next step is to take the logits of class prediction and post-process them by applying a softmax function. A penalty score is computed using a Hanning window w as follow:

$$\rho = s * (1 - \alpha) + w * \alpha \tag{5.11}$$

where ρ is the penalty score, s is the prediction score, and α the window influence. The penalty is useful to make the prediction more robust to target variations and penalize predictions that are far from the target. This score is used to find the position of the maximum value that represents the most probable region where an object could be located. As opposed to other approaches that select the most discriminative bounding box, the score is used as a center point and is checked if the score mapping falls in the bounding box region. If that is the case and the score is greater than a threshold value

	ViTCRT (updates)	ViTCRT (no updates)	Increment (%)
AUC (%)	60.38	**67.34**	+10%
P_{norm}(%)	72.54	**81.71**	+11%

Table 5.3 Comparison on OTB100

$t = 0.45$, the bounding box is used in the next run as a supplementary target allowing the tracking not only to be influenced by the first target (the most important one), but also to take advantage of the subsequent targets which brings information on the target updated state. Unfortunately, this technique has demonstrated to return lower scores (table 5.3), so it was abandoned along with classification in the tracking phase.

5.4 Fuzzy models

Our work, despite being focused mostly on visual tracking, has led to the exploration of other fields of artificial intelligence as well. During the studies on the attention mechanism and the explainability of the models, we thought of investigating the possibility of creating a deep model fully composed of fuzzy operations that could exploit the learning features used today in artificial neural networks. At the same time, given the very nature of these operations, they help us develop efficient models that may obtain results comparable with those of the current state of the art and that could be *naturally* explainable. This was possible by adapting the fuzzy compositions to local operations, made in the surroundings of the pixels of an image, similarly to convolutions. From here, it was studied an alternative layer to the convolutional one, using the MaxMin composition. All details are reported in the appendix A. As for the attention, however, it was decided to insert the same type of relationship by replacing the one that comes out of the classic matrix product, followed by the softmax function. This is because, as it has been analyzed, the type of behavior describes precisely the type of relationships that we try to express in fuzzy environment. All the details are in the appendix B. Currently, these elements are under active research and development. Finally, an integration of a transformer, based on fuzzy model, into a tracking algorithm

is also being carried out. Unfortunately, the results and framework are too premature to report in this thesis work. Results are shown in section 6.2.

Chapter 6

Experimental Results

Have no fear of perfection -
you'll never reach it

Salvador Dalì

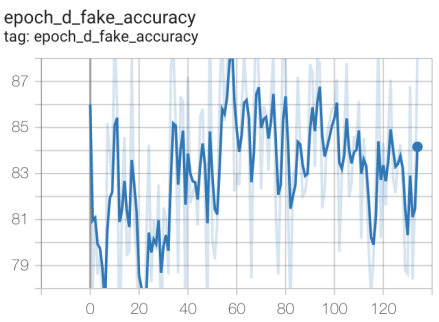
In this chapter all methodologies have been tested and compared to figure out their limitations and to motivate how any modification in architectures increases or decreases the effectiveness of theoretical reasoning. For the best model all results are reported and deeply explained.

6.0.1 Preliminary studies on GAN

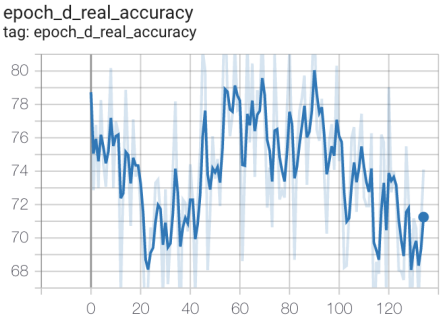
Firstly, tests are conducted on COCO test-set [49] where they have produced some promising results on static images. Before proceeding to the tracking of unknown objects, it has been attempted to specialize the network on specific tasks. The aim was to enhance the characterization ability of the network and simplify the generation process working on a single problem: the localization of a targeted face in an image. In this way, we moved from a N classes problem with potentially N different data distributions to a one class problem that is easier to work on. Model is trained on FDDB dataset [34], which provides faces location as ellipses. In figure 6.1, the learning progresses in terms of discriminator accuracy, generator IOU, and loss progression

of network discriminator/generator on face localization. It is possible to see in 6.1c that discriminator loss on training (blue curve) after an initial decreasing phase of 40 epochs, there is a convergence without big oscillations. Instead, the generator loss in 6.1d is decreasing. From the snapshot, it is not simple to define a true overfitting state, but it is presumable. It is important to remember that in GAN architecture comparing the curves of both network for good assumptions plays a key role. In figure 6.2 and figure 6.3 it is possible to check outputs on training data and validation data respectively. Network output appears quite outlined with a shape that follows the ground truth even in situations with more faces. Another situation that does not seem to interfere with the localization is the partial occlusions observable both in training (on the fourth row, with a hat that overlays the bottom right angle of the target image) and in validation (on the fourth row, where a tennis ball slightly covers the face on the top left margin of the target).

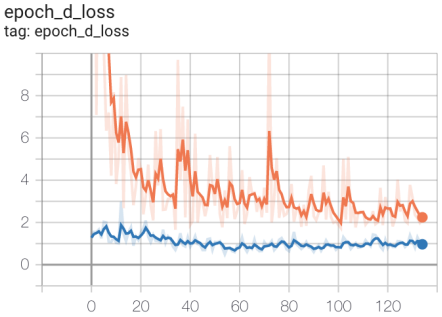
Another interesting behavior can be seen in situations where more faces are in the images: in some scenarios it seems to be able to localize a single face, but not in others. The most problematic occurrence can be found in the validation phase, where three out of four images with multiple faces are recognized in the wrong way. No post-processing is applied on the generated mask, which could surely help to reduce this kind of problem, but the aim of the methodology has been to built an end-to-end network. Clearly, it fails in the intrinsic task to generate a single object localization.



(a) D: fake accuracy

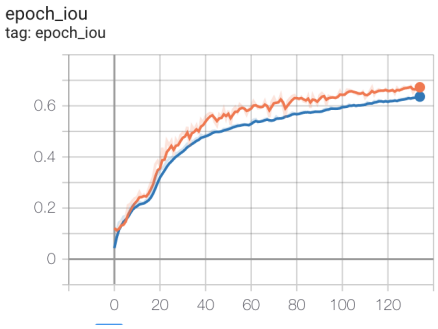


(b) D: real accuracy



(c) D: loss

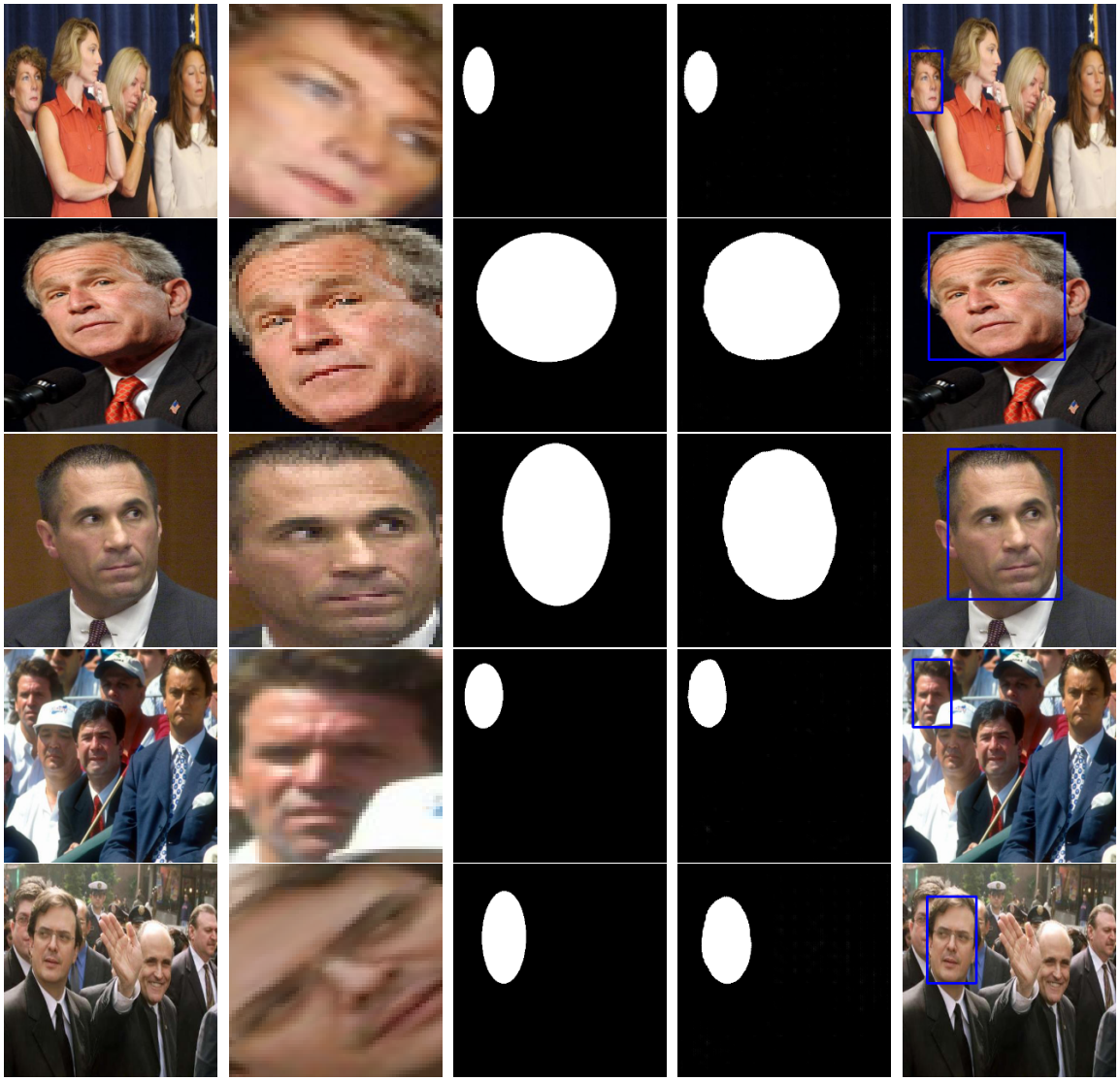
(d) G: loss



(e) G: IOU

Fig. 6.1 GAN statistics. Train ● Validation ●
Tensorboard live dashboard

It is also chosen to test the network on a single video sequence contained in OTB, where a person is moving in the scene with the camera focused on his face. The network showed problems with some kind of deformation. From various tests, it is apparent that the main problems arise with *in-out plane rotation* and changes in *illumination*.



(a) Source (b) Target (c) GT (d) Generated (e) Bounding box

Fig. 6.2 FLGAN outputs of training set

(a) Source (b) Target (c) GT (d) Generated (e) Bounding box

Fig. 6.3 FLGAN outputs of validation set

6.0.2 Siamese trackers

For each experiment in table 6.7, the test results on VOT Challenge 2019 are shown. The table shows the most relevant and difficult test in tracking. Unfortunately, none of the tests carried out has managed to reach a very important score especially considering the EAO parameter, which is the reference parameter. Some of the trackers show very fluctuating behaviors and probably structural problems within them. It is interesting to note that SiamCA-Base has achieved considerable improvement with the introduction of dynamic online learning. On the contrary, our proposal to modify it so that it could evaluate more than a single representation of the target seems not to have borne any fruit even though it indeed lowers the initial base result. The first improvement is made by using a mask generated via ode functions and along with the concatenation of high and low level features. This solution, unfortunately, involves a computational cost much higher than the sum. In addition, the dynamic layer has completely sent off the tracker using the ODE function. Another interesting behavior is the improvement that emerges when a recurrent layer encodes the two classification and regression heads. Nonetheless, the second instability is in the version that uses MCRViT, in fact, as the number of epochs increased, there was a strange spike in the performance. However, the results trend from the previous epochs seemed to improve the performance. The addition of geometric constraints and geometric online learning did not increase any results and there was no stopping difference whatsoever. Unexpected behavior occurred with the first attempts to use FCOS, a recognized state-of-the-art object detector and successfully implemented in other trackers. This is probably another sign of a problematic architecture. One of the best results (SiamSharedTransformer) was achieved by abandoning tokens within the transformers and by using internal encoding of extracted features. Among all the implementations, we chose to pursue ViTCRT both for its results, and because it manages to have a high enough performance in the training phase, processing up to 50 frames per second. This is an incredible result compared with other algorithms that took so many hours to finish even a single epoch.

	SiamFC [5]	MDNet [59]	ATOM [18]	SiamRPN++ [46]	SiamAttn [90]	TransT [13]	STARK [85]	**ViTCRT**
AUC (%)	57.1	60.6	70.3	73.3	75.2	**81.4**	**82.0**	**78.52**
P_{norm}(%)	66.3	70.5	77.1	80.0	81.7	**86.7**	**86.1**	**83.49**

Table 6.1 Results on TrackingNet test set

	SiamFC [5]	MDNet [59]	ATOM [18]	SiamRPN++ [46]	Ocean [93]	TransT [13]	STARK [85]	**ViTCRT**
AO (%)	34.8	29.9	55.6	51.7	61.1	**72.3**	**68.8**	**65.6**
$SR_{0.5}$(%)	35.3	30.3	63.4	61.6	72.1	**82.4**	**78.1**	**75.0**
$SR_{0.75}$(%)	9.8	9.9	40.2	32.5	47.3	**68.2**	**64.1**	**59.8**

Table 6.2 Results on GOT10K test set

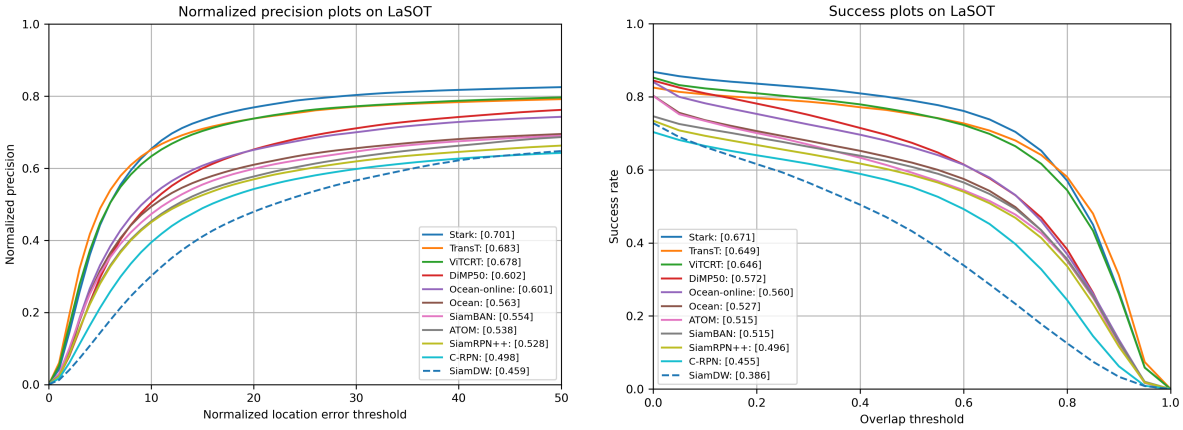
Given the promising results of ViTCRT , it was chosen to be used as the main tracker with which to carry out other tests.

In table 6.1, 6.3, and 6.2, there are results of ViTCRT on the three main benchmark large-scale datasets: TrackingNet, LaSOT and GOT10K respectively. These are compared to other state-of-the-art and reference techniques in tracking. As it can be seen, ViTCRT is not able to achieve the best results, but it reached important scores on AUC and AO metrics, which are the main comparison parameters. Moreover, in figure 6.4a and 6.4b is possible to check the normalized precision plot and the success plot compared with many state-of-art trackers. ViTCRT follows the same trend of the top trackers in literatures with same overlapping in performance and a clear distance from the others, which is indicative of higher AUC and better accuracy, error resistance, and ability to generate overlap with the true target. On TrackingNet, referencing to the evaluation server [1] appears to be in the ***Top 15*** approaches for tracking in the worldwide leaderboard. On GOT10K ViTCRT some difficulties arose, but it provided an excellent result too, as it seems to be close to STARK performance. The results have been verified on the evaluation server with a public leaderboard[2]. In the comparison ViTCRT together with TransT and STARK are the only ones to use a transformer model. Some other techniques exist, but the main reference and most powerful are

[1] http://eval.tracking-net.org/web/challenges/challenge-page/39/leaderboard/42
[2] http://got-10k.aitestunion.com/leaderboard

	SiamFC [5]	MDNet [59]	ATOM [18]	SiamRPN++ [46]	SiamAttn [90]	TransT [13]	STARK [85]	**ViTCRT**
AUC (%)	33.6	39.7	51.5	49.6	56.0	**64.9**	**67.1**	**64.57**
P_{norm}(%)	42.0	46.0	57.6	56.9	**64.8**	**73.8**	-	**67.77**

Table 6.3 Results on LaSOT test set



(a) Normalized precision plot (b) Success plot

Fig. 6.4 LaSOT comparison plots

the aforementioned. SiamFC and ATOM are very important to rely on because they brought great innovations in the tracking field. In table 6.4, there is the comparison with all other benchmark datasets.

The tracker,despite not presenting the best results yet, continues to be ranked among those with the best results and on one sequence (UAV123) it was considered with the undisputed winner of the entire current tracking scene. The proposed methodology, even though it does not represent the state of the art, is an excellent compromise in terms of performance proposal. Additionally, it represents an excellent trade-off between the ability to produce high-level tracking and execution speed, which, as we

	SiamFC [5]	RT-MDNet [59]	ATOM [18]	SiamRPN++ [46]	Ocean [93]	TransT [13]	STARK [85]	**ViTCRT**
NFS	37.7	43.3	**58.3**	57.1	49.4	**65.3**	**66.1**	**66.12**
OTB100	58.3	65.0	66.3	**68.7**	68.4	**69.5**	**68.5**	67.34
TC128	48.9	56.3	**59.9**	57.7	55.7	**59.6**	**63.1**	59.40
UAV123	46.8	52.8	63.2	59.3	57.4	**68.1**	**69.1**	**68.63**

Table 6.4 Comparison on all the remaining benchmark datasets (AUC %)

	baseline			realtime			unsupervised
	EAO	**A**	**R**	**EAO**	**A**	**R**	**AUC**
ViTCRT	0.433	0.774	0.711	0.434	0.774	0.711	0.609

Table 6.5 Preliminary results of ViTCRT on VOT2022 - Short Term - Bounding Box challenge

Tracker	**Speed (FPS)**	**FLOPs (G)**	**Params (M)**
ViTCRT ★	83.7	11.3	18.8
STARK-S50★	42.2	12.1	28.1
STARK-ST50★	41.8	12.8	28.2
TransT-N2♦	70.0	-	16.7
TransT-N4♦	50.0	-	23.0
SiamRPN++★	35.0	48.9	54.0

Table 6.6 Comparison with major trackers, especially with the two that use Transformers. ★ tested on Nvidia Tesla V100. ♦ tested on Nvidia Titan RTX

.

see from Table 6.6, is able to reach 80 FPS in its basic configuration, preserving a relatively low number of parameters. The results presented were achieved with the hardware configuration specified in the table.

Tracker Identifier	**A** (↑)	**R** (↓)	**LN**	**EAO** (↑)
SiamCA-Base	0.562	0.843	168.0	0.204
SiamCA-Base-ODYN	0.558	0.727	145.0	0.226
SiamCA-Base-ODYNM	0.556	0.803	160.0	0.201
SiamCA-ODE-SUM	0.562	0.843	168.0	0.204
SiamCA-ODE-SUM-ODYN	0.146	5.006	998.0	0.018
SiamCA-ODE-CAT	0.556	0.778	155.0	0.229
SiamSPL-ODE-SUM (ep. 20)	0.487	0.848	169.0	0.194
SiamSPL-ODE-SUM (ep. 25)	0.528	0.873	174.0	0.195
SiamSPL-ODE-SUM (ep. 40)	0.538	0.873	174.0	0.207
SiamSPL-ODE-SUM (ep. 45)	0.503	1.018	203.0	0.180
SiamCA-ODE-CAT-ATTN	0.544	0.762	152.0	0.209
SiamCA-ODE-SUM-ATTN-RCT	0.580	0.783	156.0	0.218
SiamCA-ODE-SUM-ATTN-RCT-ODYN	0.398	0.762	152.0	0.153
SiamCA-ODE-SUM-ATTN-ViT (ep. 25)	0.586	0.888	177.0	0.197
SiamCA-ODE-SUM-ATTN-ViT (ep. 27)	0.588	0.958	191.0	0.191
SiamCA-ODE-ATTN-MCRViT (ep. 15)	0.550	0.903	180.0	0.191
SiamCA-ODE-ATTN-MCRViT (ep. 20)	0.569	0.903	180.0	0.195
SiamCA-ODE-ATTN-MCRViT (ep. 25)	0.571	0.933	186.0	0.195
SiamCA-ODE-ATTN-MCRViT (ep. 28)	0.514	3.045	607.0	0.076
SiamCA-ATTN-MCRViT (ep. 15)	0.588	0.913	182.0	0.205
SiamCA-ATTN-MCRViT (ep. 20)	0.585	0.933	186.0	0.205
SiamCA-ATTN-MCRViT (ep. 25)	0.563	0.818	163.0	0.213
SiamCA-ATTN-MCRViT (ep. 28)	0.584	0.918	183.0	0.207
SiamCA-ATTN-MCRViT-MTL	0.583	0.838	167.0	0.201
SiamSPL-ATTN-MCRViT	0.584	0.903	180.0	0.196
SiamSPL-VTGC (ep. 13)	0.570	0.878	175.0	0.201
SiamSPL-VTGC (ep. 15)	0.515	1.084	216.0	0.168
SiamSPL-VTGC (ep. 20)	0.527	1.084	216.0	0.161
SiamSPL-VTGC-OAE	0.515	1.084	216.0	0.168
SiamSPL-VTGC-MTL	0.412	0.998	199.0	0.162
SiamSPL-VTGC-MTL-OAE	0.421	0.998	199.0	0.162
SiamSPL-GC-FCOS	0.412	1.866	372.0	0.101
SiamSPL-VTGC-FCOS	0.149	1.209	241.0	0.084
SiamConnector-MCRCeiXt	0.487	1.971	393.0	0.103
SiamSharedTransformer	0.588	0.597	119.0	0.271
ViTCRT	**0.784**	**0.445**	**87.0**	**0.398**

Table 6.7 Results of trackers on VOT2019. Caption in table 5.1.
(**A**) Accuracy; (**R**) Robustness[3]; (**LN**) Lost Numbers; (**EAO**) Expected Average Overlap

[3]Up to VOT2019 robustness is computed as lower is better.

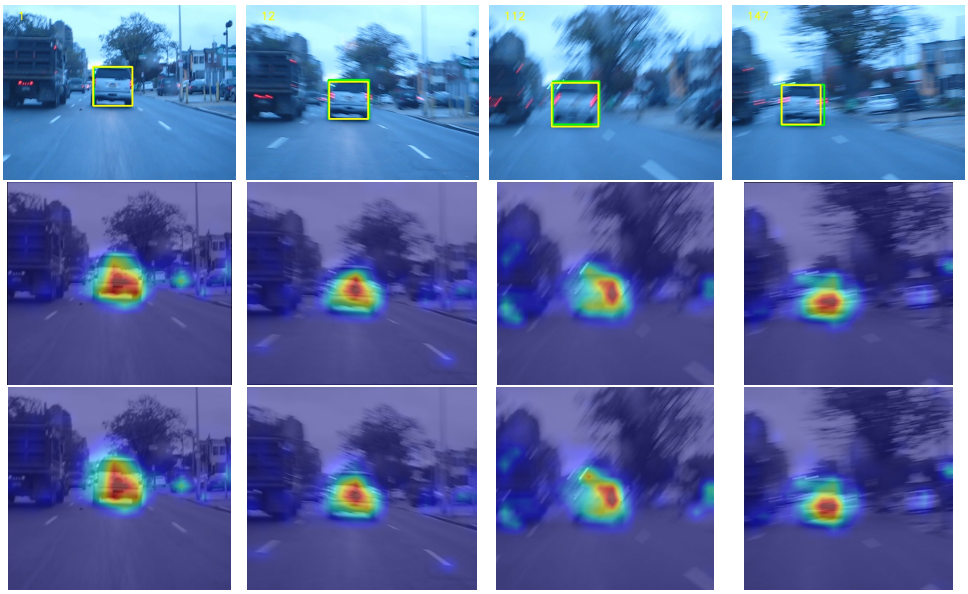


Fig. 6.5 Good outputs for sequence *BlurCar1* in OTB100 with the attention of two tokens overlaying the image. *First row*: Bounding Box outputs. *Second row*: Regression token attention. *Third row*: Classification token attention.

6.1 Tracking Attention

Given the use of a transformer and that it relies primarily on the attention mechanism, it is interesting for the purposes of the research and explanation of the results to see where attention is focused on the image. To carry out the proposed tests we used video sequences from the test dataset and to verify the contribution of attention we considered the procedure of attention rollout [1] that allows to extract, through a combination of attention in each layer, which is the area of the image where the correlation of the input is maximum. In the basic case, it allows us to extract where the algorithm is more focused, but in the proposed case, given the type of input or the concatenation of target and search image, it is interesting to see how both of the learned tokens behave. In figure 6.5, it is possible to see on the first row, the output bounding box (yellow) compared with the ground truth (red) and on the second and third columns, how the attention is mapped on the image, taking into account both tokens. It seems that the tracker is able to correctly recognize the subject even in the presence of not excessive blurring, showing a good robustness for this kind of visual attribute. In the

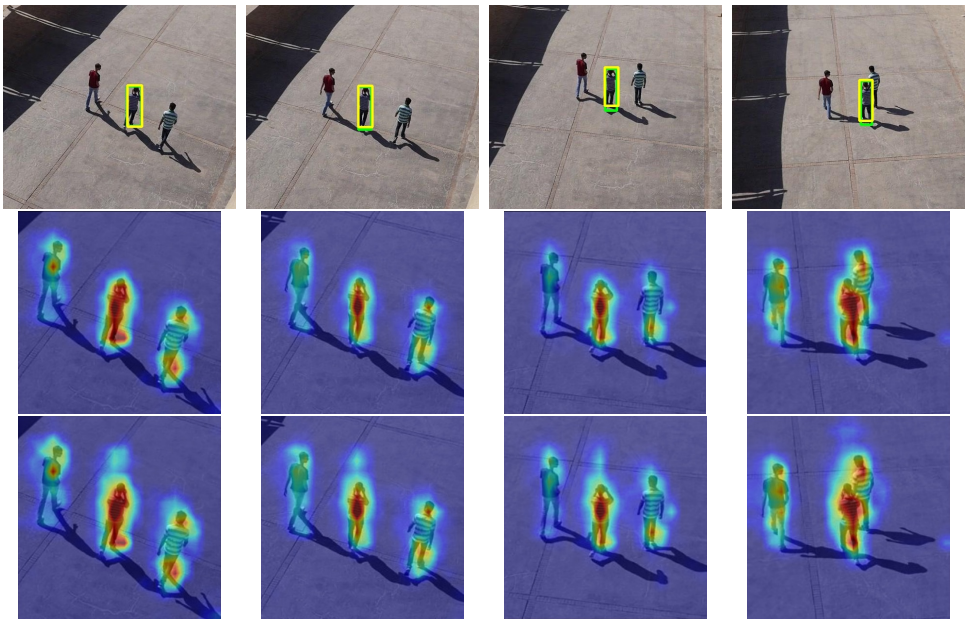


Fig. 6.6 Good outputs for sequence *Group1_2* in UAV123 with the attention of two tokens overlaying the image. *First row*: Bounding Box outputs. *Second row*: Regression token attention. *Third row*: Classification token attention.

same way, it is possible to observe in the figure 6.6 a second sequence (the image has been cropped to improve the visualization), in which the shooting is done by a drone that simultaneously rotates around the subject and moves away, leading to a change in perspective and a variation in scale. At the same time, we see the person on the right of the subject being tracked looking more and more like the target subject and moving in front of it. The tracking continues to occur successfully and the attention is at first completely focused on the target while with the transit of a similar subject, it slightly shifts towards it, but this is not enough to confuse the network.

In figure 6.7, there are problematic frames from both sequences. In the first two columns, the attention, due to the movement of the camera and the consequent extreme blur effect expands its area of importance beyond the designated object. The subject is encapsulated in the bounding box, but noisy objects entered too. On the last two columns, it is observable how the network gets confused and the attention starts to be very relevant for the person on the left; the bounding box follows this behavior, expanding itself to both subjects and when most of the attention goes to the wrong

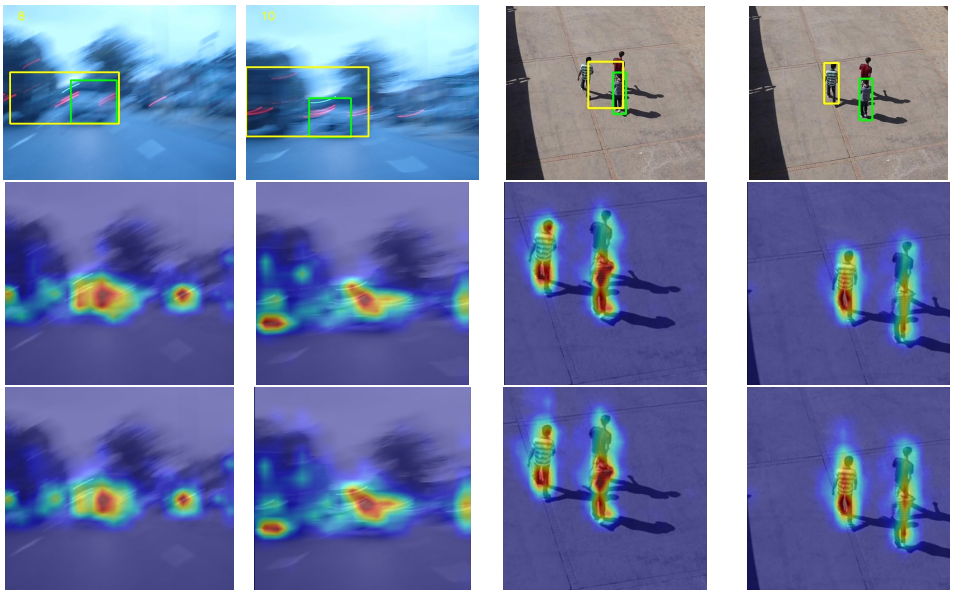


Fig. 6.7 Bad outputs for both sequences with the attention of two tokens mapped on the image. *First row*: Bounding Box outputs. *Second row*: Regression token attention. *Third row*: Classification token attention.

person, the bounding box too is encouraged to follow it. Those visualizations seem to demonstrate that attention, for the most part, follows the behavior of the bounding box, and vice-versa, as expected. Therefore, it is essential to specify that an uncertainty in the attention brings the uncertainty in the bounding box and class prediction. Furthermore, it is possible to notice how the attention of the two tokens is really similar to each other. For this reason, it can be hypothesized that in the training process, the two tokens undergo a similar influence regarding the correlation process. This could lead to suppose that the influence of the multiple loss functions also propagates into the other elements, as it was expected to happen.

	MNIST				CIFAR10			
	T		V		T		V	
Model	**A**	**L**	**A**	**L**	**A**	**L**	**A**	**L**
Conv2D+ReLU	0.99	0.007	0.97	0.1	**0.59**	1.17	**0.53**	1.29
MaxMin	0.99	0.008	0.97	0.08	**0.52**	1.38	**0.49**	1.46
MaxMin (2L)	0.99	7.9e-4	0.98	0.09	**0.52**	1.36	**0.50**	1.43

Table 6.8 Results comparison

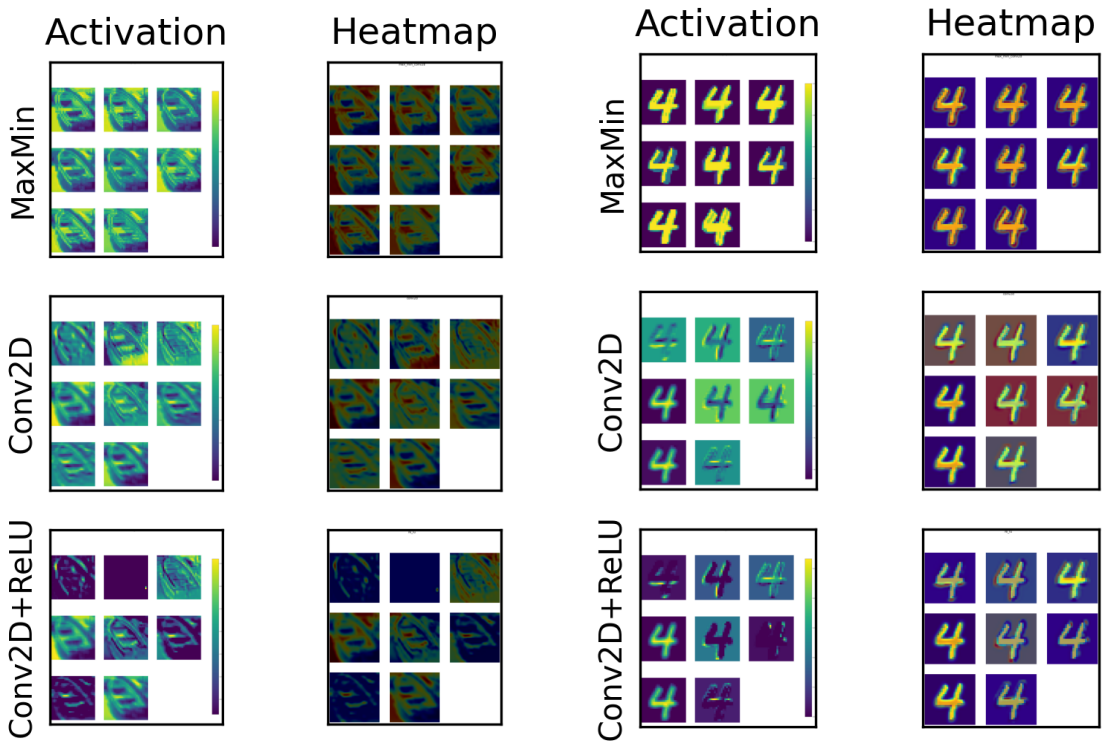
6.2 Fuzzy Models

In this section, results will be shown on the two studied fuzzy methodologies with an evaluation from an explainability standpoint. The techniques are still under research, so we will show a comparison with traditional techniques and the results, which already look very promising.

6.2.1 Advanced Fuzzy Relational Neural Network

AFRNN has been applied to images classification and the MNIST [44] and CIFAR10 [43] datasets are considered for this study. Input images are scaled in range $[0-1]$ as fuzzification step. The single hidden layer architecture uses a feature map of size 8 while in the two hidden layers setup there are respectively feature maps of size 8 and 16 respectively. Weights are randomly initialized using a uniform distribution in range $[0-1]$ in order to define a random degree order. Further weights are constrained to be in the same range after the backpropagation phase, because they have to define a data membership degree for all channels at any moment. There is a hard clipping of the weights on boundaries, like in the gradient clipping case. All layers have a kernel size of 3 on spatial dimensions. In table 6.8 it is possible to notice performance of CNN compared with fuzzy architectures. It is further possible to observe that performances on MNIST are comparable, but on CIFAR10, CNN is slightly better than AFRNN. However, observing the activations and the heatmaps (Fig. 6.8) of the models and some others visualizations based on GradCAM [75], Gradients*Inputs [4], and Integrated Gradients [62] (Fig. 6.9, 6.10), it is plain to see that AFRNN can

explain the information used in classification more accurately. GradCAM shows that conv2d and relu function cut-out important features of the object, retaining the non relevant one. Instead, AFRNN, if irrelevant areas are present, is able to preserve the shape of the ships. This appear clear from observing the gradients. Both techniques have some noise, but the fuzzy module is able to focus more on the image subject. The same analysis is possible with MNIST in fig. 6.10, where gradients, that is the attention of the network, are located following the object shape. This statement is not valid for the convolutional layer that shows a lot of noise and therefore it is unable to focus on the main subject.



(a) CIFAR10 (b) MNIST

Fig. 6.8 Activations and heatmaps of layers on CIFAR10 and MNIST datasets

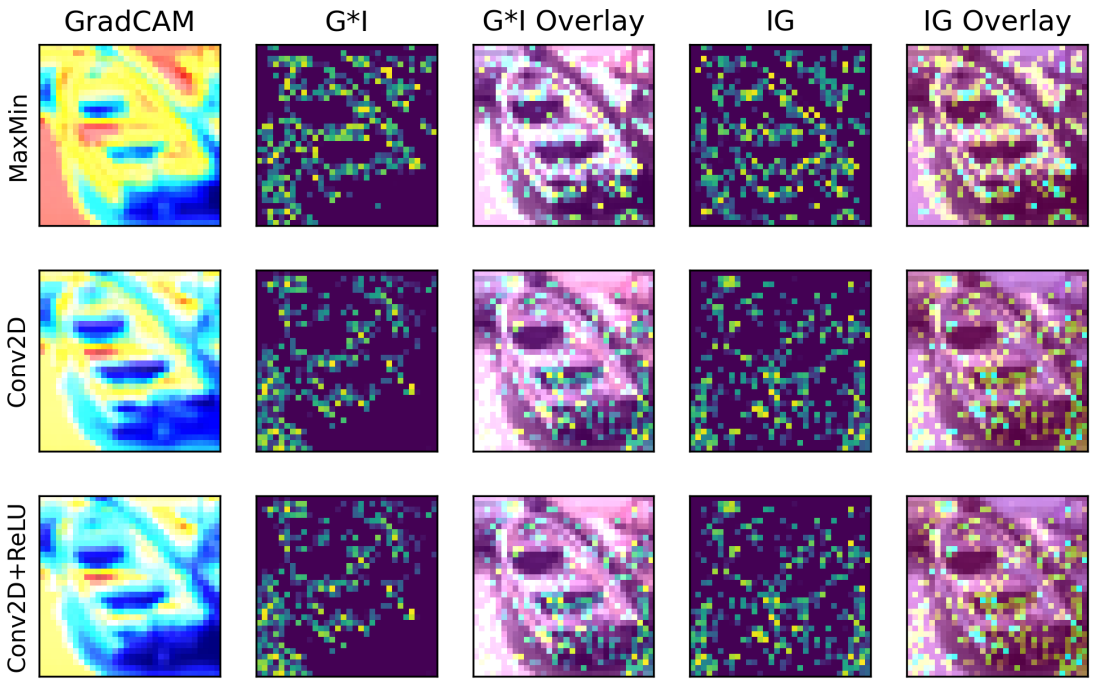


Fig. 6.9 Comparison between MaxMin layer and Conv2D network on CIFAR10. G*I is Gradients*Inputs - IG is Integrated Gradients

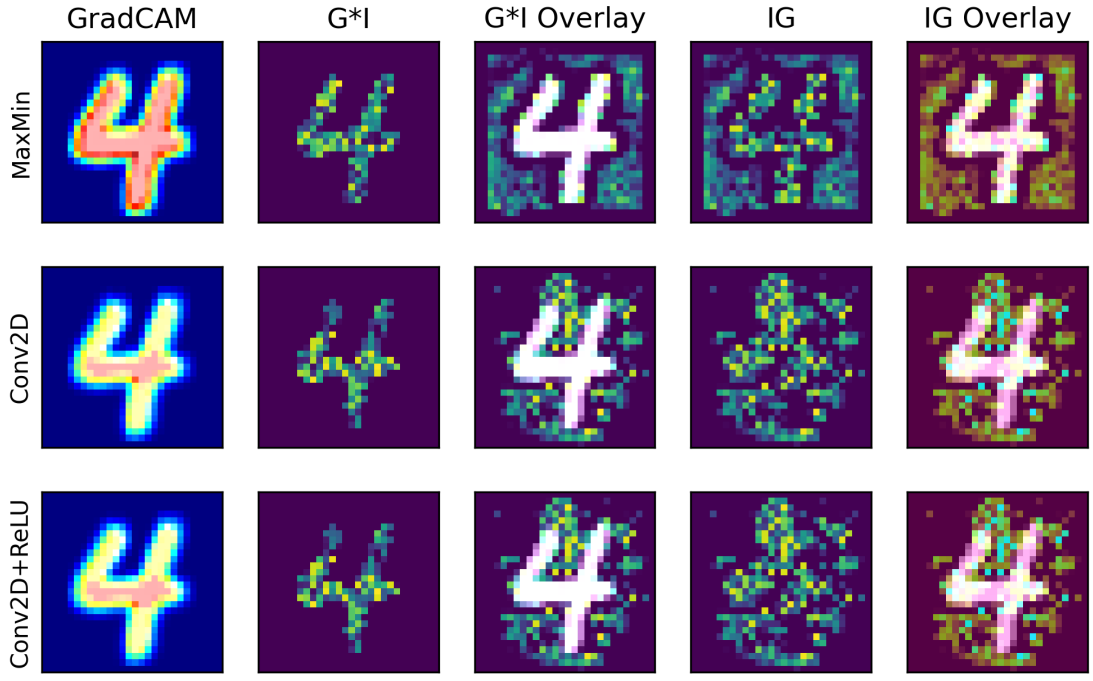


Fig. 6.10 Comparison between MaxMin layer and Conv2D network on MNIST. G*I is Gradients*Inputs - IG is Integrated Gradients

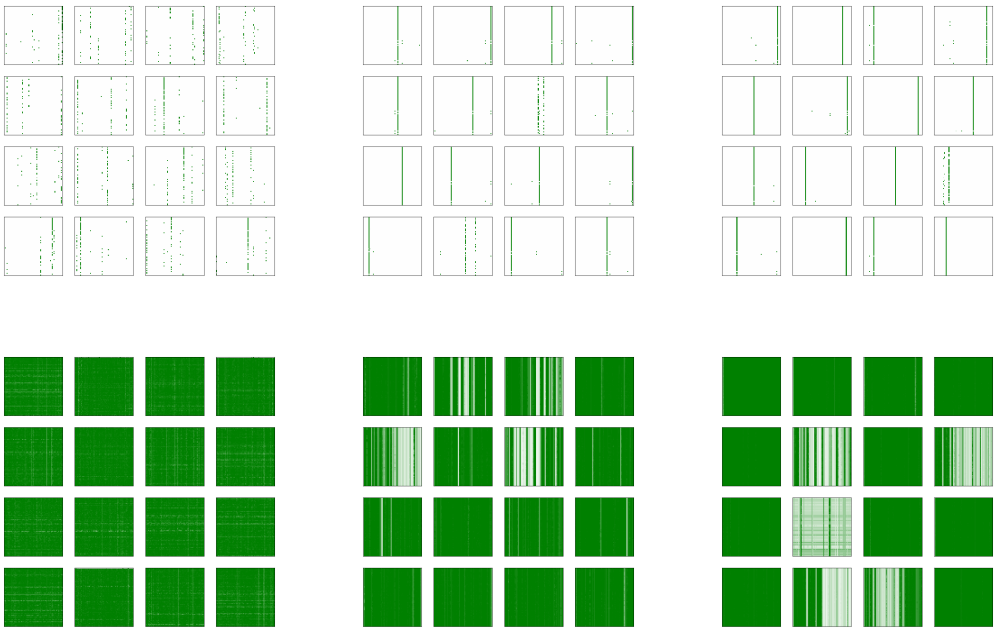
	Patch size	Embed	# Encoder	Heads	FFN	Dropouts	Comp	Memb
ViT	4	1024	8	16	2048	0.1	-	
Fuzzy ViT-1	4	1024	8	16	2048	0.1	MaxMin	Gaussian
Fuzzy ViT-2	4	1024	8	16	2048	0.1	MaxMin	Sigmoidal

Table 6.9 Composition of ViT regular and fuzzy versions

6.2.2 Fuzzy Transformer

For this study the transformer model chosen for the experiment is a *standard* ViT [20], composed of 8 encoder layers, 16 heads, an embedding dimension of 1024 elements and a patch size of 4. All parameters are reported in table 6.9, trained on CIFAR10 with Adam [36] optimizer and cosine learning decay. For the fuzzy alternative, the same settings are used, but the Multihead Attention has been replaced with the Fuzzy Attention and with Max-Min composition, using two different fuzzification functions, the sigmoidal membership and the Gaussian membership. The fuzzy model is able to reach an accuracy of around 60% after few epochs without pretraining while the basic ViT needs be trained a lot before convergence.

A first short comparison can be done on the attention heads visualization as shown in figure 6.11. In the first row, the quantity of information presented in each head is very little and it helps to focus on the most important elements during the forward phase. On the contrary, in the second row, it is possible to notice that the amount of information is very high and going on with the number of layers, some lose a relevant quantity of information in order to provide only the most important ones, others, on the other hand, appear to be completely dense. The situation is similar for both Fuzzy ViT-1 and Fuzzy ViT-2, so in the figure, the Fuzzy ViT-1 experiment is shown. This leads to a problem that needs to be addressed in the fuzzy composition, because by doing so, it does not take into account the relationships that are not helpful for learning the network. Research on this avenue, as a consequence, will move forward, trying to reduce this problem. The first idea that we are trying to prune the heads that carry too much information from the entropy computation. In this way, it is possible to eliminate those that have too high entropy, as it exceeds a predefined threshold value.



(a) Layer 1 (b) Layer 5 (c) Layer 8

Fig. 6.11 Attention heads at third encoder. The *first row* contains attentions of the standard method. The *second row* contains the attentions of the fuzzy version

This leads to the need to analyze the focus from an explainability perspective. For this purpose, it has been chosen to use the rollout [1] method on CIFAR10 test set, verifying the correlation of the classification token over the input image. In figure 6.12, on the first columns, there are the input images, whereas on the second row, the rollout of ViT is located and on the third and fourth the rollouts of Fuzzy ViT-1 and Fuzzy ViT-2 can be found respectively. There is a strong tendency of the basic attention not to find coincident correlations with the input images, but is confused by the background elements, such as in the case of the frog, which has a very similar color to that of the background or the car, where the tones of the wheels share similarities to those of the asphalt. On the other hand, we can see that the fuzzy attention, although it did not seem to give a good result from the point of view of filtering information, as discussed above, it takes more into account those which are the relationships with objects. Eventually, it is apparent how the memberships influence the final result.

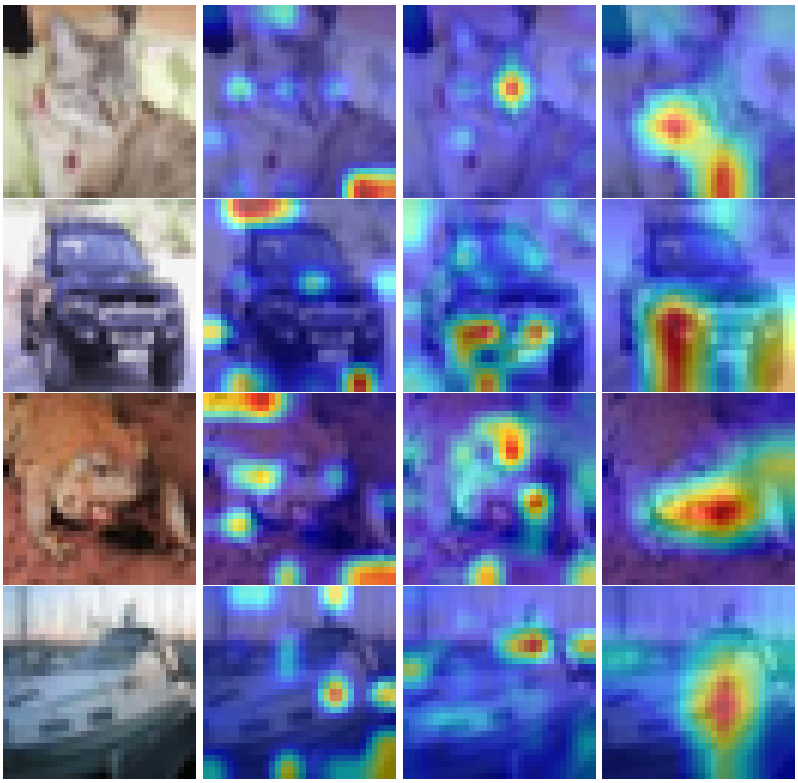


Fig. 6.12 Attention rollout for each model on four samples of CIFAR10 test set

Fuzzy ViT-1 tends to focus on very specific elements, such as the cat's face, the bumper, and the car wheels. Despite being compressed, this produces a focus that easily defines the object and its absolute position. In addition, in this case we can witness instances where the background has created non-existent correlations, such as with the frog. The last Fuzzy ViT-2 experiment retains the properties of the previous one, but there is a greater dilation, or rather, correlations in a larger area on the object itself without, however, having a high percentage of focus in wrong areas of the image.

Conclusions

Plausible nonsense is always better than a possibility that doesn't convince.

Aristotle

6.3 Conclusions and future works

The dissertation presented herein is a study of visual object tracking and new techniques that may or may not be useful in the construction of new methodologies. Various techniques have been developed and shown, inspired by the state-of-the-art approaches, which have made it possible to outline a path towards an innovative solution in line with the current interest of the scientific community. This solution, named ViTCRT , possesses the methodological characteristics at which we have aimed, being internally composed of a transformer that manages to be used without difficulty on images and with a fair amount of data in a complex task and not immediate from the architectural standpoint. Finally, this has allowed to find an effective solution for the evaluation of result, both from the point of view of metrics, and from that of computational resources, being today of great importance, given the extreme power of deep learning methodologies, which is counterbalanced with the ever-increasing demand for performing physical resources. In addition, such a solution enables to compare itself with the state of the art and to compete with it, so much so that it can be included among the highest positions in the world leaderboards (which are constantly updated

by the community) and in the VOTChallenge competition, having objectively become the fundamental point of arrival in the world of tracking and decreeing, thanks to the tools provided and the type of evaluations that are directly carried out by those responsible for the race, who are the best trackers in the world.

Overall, the following work proved an inspiration to conduct research that has brought different aspects of machine learning and computer vision closer together. The explainability results showed from the tracker also allows to judge the quality of the process, not just from a numerical point of view, but from a human perspective as well. Nowadays, it is crucial to demonstrate why an artificial intelligence algorithm is working in such a way. Thanks to our study, we have shown that ViTCRT can be described as an algorithm whose predictions are trustworthy because it has provided, through attention analysis, a high degree of consistency whether it returns a good result or not. From this, it is natural to think about how to continue the work done with the possibility of extracting new ideas and present new research proposals that can push the research even further. In part this has already happened during the research activity, where the inspiration to build alternative systems to the conventional ones has risen, thanks to fuzzy logic that allows to have a good degree of reliability, having analyzed in depth the answer that these techniques have returned. This has been possible in every instance through the use of explainability techniques and even though in some cases, like in very complex models (such as fuzzy transformer), the response was improvable, we witnessed that the models at the base have this type of property that makes us confident about the next results. In the future, we expect to be able to build more efficient tracking systems and thanks to XAI systems, this can be easily achieved, making the process leading to improvement even simpler, intervening with problem identification and allowing an extensive dissemination of these methodologies.

6.4 Acknowledgments

This work was completed with resources provided by the University of Naples "Parthenope", Department of Science and Technologies, Research Computing Facilities[4]. Without *PurpleJeans*, no researchs, no testing, nothing done in these years would have been possible.

[4] https://rcf.uniparthenope.it

Appendix A

Advanced Fuzzy Relational Neural Network

In the following sections, a fuzzy neural model will be shown, named Advanced Fuzzy Relational Neural Network (AFRNN), which is proposed as an alternative to convolutional layer, from performance and explainability perspective.

Introduction and related works

Fuzzy systems have been used to model the smoothness in classic problems. They have produced great results during the year and they are actively under research. Nowadays, most of the researchers have decided to study artificial neural networks and one of the most used operations is the convolutional one. Over the years, many neuro-fuzzy systems have appeared, but with less relevance. The aim is to model a new neuro-fuzzy system with a set of operations that are based on the application of t-norm locally, in particular on a patch sliding windows fashion, in order to extrapolate relevant information at a higher level of abstraction. Furthermore, convolutional operations need post-hoc techniques for improving their interpretability and explainability. In the last years, fuzzy systems have raised great interest for their ability to develop reliable and explainable systems. This work aims to introduce a fuzzy relational neural

network based on a model for extrapolating relevant information from images data permitting to obtain a clearer indication on the classification processes. Techniques for eXplainable Artificial Iintelligent (XAI) can be model agnostic (i.e. they can be applied to any AI algorithm), or model specific (i.e. they can only be applied to a specific AI algorithm). Moreover, they can be ante-hoc (transparent or "white box/glass box" approaches, explainable by design or inherently explainable) or post-hoc (divided into global explanations or local explanations) explainability methods [37]. Ante-hoc methods are explainable by design and great interest in the last years has been showed in Fuzzy Rule-based systems [8, 54, 55]. In this work we propose a fuzzy relational neural network that is based on a fuzzy inference scheme for the classification of images.

Triangular Norm

A *Triangular Norm* (T-norm) generalizes intersection in a lattice and conjunction in logic. The name "triangular norm" refers to the fact that in the framework of probabilistic metric spaces, t-norms are used to generalize the triangle inequality of ordinary metric spaces. A t-norm is a function $T : [0,1] \times [0,1] \to [0,1]$ that satisfies many properties:

- Commutativity: $\top(a,b) = \top(b,a)$
- Monotonicity: $\top(a,b) \leq \top(c,d)$ if $a \leq c$ and $b \leq d$
- Associativity: $\top(a, \top(b,c)) = \top(\top(a,b),c)$
- Identity element: $\top(a,1) = a$

T-norms are used to model the intersection of fuzzy sets or as aggregation operators. Many t-norms have been developed during the years, some of the most famous ones are:

- Gödel t-norm (minimum): $\top_{\min}(a,b) = \min\{a,b\}$

- Product t-norm: $\top_{prod}(a, b) = \max\{a \cdot b\}$
- Łukasiewicz t-norm: $\top_{Luk} = \max\{0, a + b - 1\}$

Another operator that is associated with t-norms is the t-conorms. Given a t-norm $\top$, the complementary conorm is defined by:

$$\perp(a, b) = 1 - \top(1 - a, 1 - b) \tag{A.1}$$

This also generalizes the *De Morgan's laws*.

Referring to the previous t-norm, the corresponding t-conorms are:

- Maximum t-conorm: $\perp_{\max}(a, b) = \max\{a, b\}$ dual to Gödel t-norm
- Probabilistic sum: $\perp sum(a, b) = a + b - a \cdot b$ dual to product t-norm
- Bounded sum: $\perp_{Luk}(a, b) = \min\{a + b, 1\}$

Fuzzy Relational Neural Network model

Fuzzy Rule-based Systems (FRSs), have raised great interest in XAI in the last years as ante-hoc methodologies [37]. The main components of any FRS are the knowledge base (KB) and the inference engine module. The KB comprises all the fuzzy rules within a rule base (RB) and the definition of the fuzzy sets in the data base. The inference engine includes a fuzzification interface, an inference system, and a defuzzification interface [8, 15]. Fuzzy Relational Neural Network (FRNN) [16] is an adaptive model based on a FRS. AFRNN can be developed with different norms and a backpropagation algorithm is used for learning. In this work, we have modeled local t-norms, modifying the inner operation of convolution and replacing the linear combination provided by matrix multiplication with fuzzy operators. We have defined a receptive field that applies a triangular operation to a restricted area. As it usually happens in convolution, we have a kernel size of $N \times M \times C_{in} \times C_{out}$ where N and M are the spatial dimensions, C_{in} is the number of input channels and C_{out} the number of output features maps.

Kernel slides over the image with a parametric step. Weights are initialized in range $[0, 1]$ and constrained to be in the same interval after the optimization step, using a weight clipping operation. After some studies, this has proved to be the best method, because by means of a scaling operation (such as minmax scaling or Gaussian projection) weights are forced to endure a variation that, according to a determined logic, completely affects all the learned values. This implies that the net has to re-learn all the values, putting a halt to network so as to advance in learning. With clipping, instead, it automatically learns to take the weights in the desired numeric interval after few epochs.

The network structure is composed of an input layer and a fuzzification layer, where the membership function is just a scaling of the pixel value in range $[0 - 1]$. We compare the results by using one or two hidden layers. Next, there is a defuzzification operation that consists of a fully connected layer like in [16] and an output layer with a Categorical Crossentropy is used for classification. Architectures have been tested with and without a threshold activation function and a modification of leaky relu with a minimum boundary > 0. Networks are compared with equivalent CNN architectures.

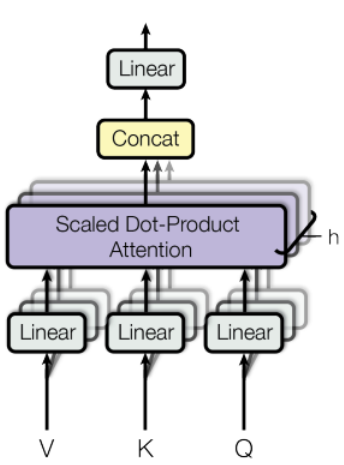
Appendix B

Fuzzy Attention in Transformers

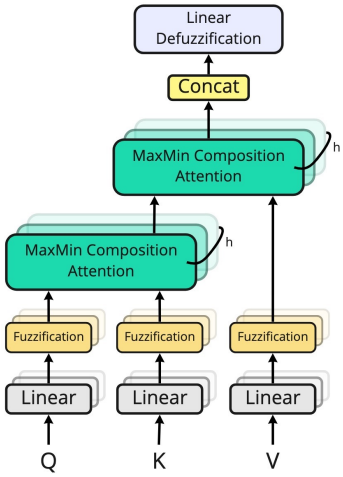
The most effective mechanism in Transformers is the Multihead Attention (figure B.1a. It is continuously under study and a lot of variants have been made. Mathematically, it is expressed as:

$$\text{MHA}(q,k,v) = softmax\left(\frac{qk}{\sqrt{d}}\right)v \tag{B.1}$$

Where q, k, v are the *query*, *key*, *value* matrices respectively, which can be represented as $(q = k = v)$ for self-attention or $(q = v, k)$ for cross-attention. Softmax function with its argument is the attention matrix and the multiplication with v gives the attention applied to it. During the research activity, the deep study of the attention made evident that attention can be seen as a composition function of two matrices. Usually, this composition is employed when the common matrix multiplication is followed by a softmax function applied on each row. Obviously, the matrix multiplication can be seen as a linear combination that returns the relation of every row element with each column element. This highlights to the importance of finding the correlation of every element with each other. On the other hand, the softmax function, points out what the most important correlations for each element in a row are, because its nature is to normalize each element, lowering the less significant components and raising up the most relevant ones. This allows to obtain the self or cross attention of the



(a) Classical Multihead Attention



(b) Fuzzy Multihead Attention

matrices, which could further be seen as an *importance mask* and a subsequent matrix multiplication of the attention for the v matrix. The aforementioned are the elements on which it is important to focus in the output projection.

Starting from the concept of composition and the concept of fuzzy composition expressed by t-norms (refers to appendix A), a transformer that uses Fuzzy Attention is built. The modification, shown in figure B.1b, is done by replacing the attention multiplication with a max-min t-norm. As a result, the softmax function is removed because the same kind of relation is intrinsically represented by the t-norm. In this case, the attention is described as:

$$\text{attn}(q, k) = \max \min(q, k) \tag{B.2}$$

The last multiplication is another max-min composition.

$$\text{Fuzzy\_MHA}(q, k, v) = \max \min(\text{attn}(q, k), v) \tag{B.3}$$

Based on the fuzzy logic needs, the inputs q, k, v must be fuzzified before they can be processed with a fuzzy operation; for the membership function, sigmoidal membership and a Gaussian membership functions are tested on the linear projection of the inputs. Additionally the attention itself is defuzzified by using the final projection that does not

change from the original one in attention because it naturally follows the defuzzification rule in [16].

Given the previous work on computer vision and the ease of explaining data in the case of images, fuzzy attention was applied to the Vision Transformer, which is, moreover, the main subject of study in the proposed work. It is also possible to easily verify the contribution of the classification token by using attention rollout to see what the most focused element is in the image.

References

[1] Abnar, S. and Zuidema, W. (2020). Quantifying attention flow in transformers. *arXiv preprint arXiv:2005.00928*.

[2] Arulampalam, M. S., Maskell, S., Gordon, N., and Clapp, T. (2002). A tutorial on particle filters for online nonlinear/non-gaussian bayesian tracking. *IEEE Transactions on signal processing*, 50(2):174–188.

[3] Babenko, B., Yang, M.-H., and Belongie, S. (2009). Visual tracking with online multiple instance learning. In *2009 IEEE Conference on computer vision and Pattern Recognition*, pages 983–990. IEEE.

[4] Baehrens, D., Schroeter, T., Harmeling, S., Kawanabe, M., Hansen, K., and Müller, K.-R. (2010). How to explain individual classification decisions. *The Journal of Machine Learning Research*, 11:1803–1831.

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

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

[7] Briechle, K. and Hanebeck, U. D. (2001). Template matching using fast normalized cross correlation. In *Optical Pattern Recognition XII*, volume 4387, pages 95–102. International Society for Optics and Photonics.

[8] Camastra, F., Ciaramella, A., Giovannelli, V., Lener, M., Rastelli, V., Staiano, A., Staiano, G., and Starace, A. (2015). A fuzzy decision system for genetically modified plant environmental risk assessment using mamdani inference. *Expert Systems with Applications*, 42(3):1710–1716.

[9] Carion, N., Massa, F., Synnaeve, G., Usunier, N., Kirillov, A., and Zagoruyko, S. (2020). End-to-end object detection with transformers. In *European conference on computer vision*, pages 213–229. Springer.

[10] Čehovin, L. (2017). Trax: the visual tracking exchange protocol and library. *Neurocomputing*, 260:5–8.

[11] Chen, R. T., Rubanova, Y., Bettencourt, J., and Duvenaud, D. K. (2018). Neural ordinary differential equations. *Advances in neural information processing systems*, 31.

[12] Chen, X., Yan, B., Zhu, J., Wang, D., Yang, X., and Lu, H. (2021a). Transformer tracking. In *CVPR*.

[13] Chen, X., Yan, B., Zhu, J., Wang, D., Yang, X., and Lu, H. (2021b). Transformer tracking. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 8126–8135.

[14] Cho, K., Van Merriënboer, B., Gulcehre, C., Bahdanau, D., Bougares, F., Schwenk, H., and Bengio, Y. (2014). Learning phrase representations using rnn encoder-decoder for statistical machine translation. *arXiv preprint arXiv:1406.1078*.

[15] Ciaramella, A., Tagliaferri, R., Pedrycz, W., and Di Nola, A. (2006a). Fuzzy relational neural network. *International Journal of Approximate Reasoning*, 41(2):146–163.

[16] Ciaramella, A., Tagliaferri, R., Pedrycz, W., and Di Nola, A. (2006b). Fuzzy relational neural network. *International Journal of Approximate Reasoning*, 41(2):146–163.

[17] Comaniciu, D., Ramesh, V., and Meer, P. (2000). Real-time tracking of non-rigid objects using mean shift. In *Proceedings IEEE Conference on Computer Vision and Pattern Recognition. CVPR 2000 (Cat. No. PR00662)*, volume 2, pages 142–149. IEEE.

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

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

[20] Dosovitskiy, A., Beyer, L., Kolesnikov, A., Weissenborn, D., Zhai, X., Unterthiner, T., Dehghani, M., Minderer, M., Heigold, G., Gelly, S., et al. (2020). An image is worth 16x16 words: Transformers for image recognition at scale. *arXiv preprint arXiv:2010.11929*.

[21] Everingham, M., Eslami, S. M. A., Van Gool, L., Williams, C. K. I., Winn, J., and Zisserman, A. (2015). The pascal visual object classes challenge: A retrospective. *International Journal of Computer Vision*, 111(1):98–136.

[22] Fan, H., Lin, L., Yang, F., Chu, P., Deng, G., Yu, S., Bai, H., Xu, Y., Liao, C., and Ling, H. (2019). Lasot: A high-quality benchmark for large-scale single object tracking. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 5374–5383.

[23] Galoogahi, H. K., Fagg, A., Huang, C., Ramanan, D., and Lucey, S. (2017). Need for speed: A benchmark for higher frame rate object tracking. *arXiv preprint arXiv:1703.05884*.

[24] Gholami, A., Keutzer, K., and Biros, G. (2019). Anode: Unconditionally accurate memory-efficient gradients for neural odes. *arXiv preprint arXiv:1902.10298*.

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

[26] Goodfellow, I., Pouget-Abadie, J., Mirza, M., Xu, B., Warde-Farley, D., Ozair, S., Courville, A., and Bengio, Y. (2014). Generative adversarial nets. *Advances in neural information processing systems*, 27.

[27] Guo, D., Wang, J., Cui, Y., Wang, Z., and Chen, S. (2020). Siamcar: Siamese fully convolutional classification and regression for visual tracking. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 6269–6277.

[28] Guo, Q., Feng, W., Zhou, C., Huang, R., Wan, L., and Wang, S. (2017). Learning dynamic siamese network for visual object tracking. In *Proceedings of the IEEE international conference on computer vision*, pages 1763–1771.

[29] He, D.-C. and Wang, L. (1990). Texture unit, texture spectrum, and texture analysis. *IEEE transactions on Geoscience and Remote Sensing*, 28(4):509–512.

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

[31] Henriques, J. F., Caseiro, R., Martins, P., and Batista, J. (2014). High-speed tracking with kernelized correlation filters. *IEEE transactions on pattern analysis and machine intelligence*, 37(3):583–596.

[32] Huang, L., Zhao, X., and Huang, K. (2019). Got-10k: A large high-diversity benchmark for generic object tracking in the wild. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 43(5):1562–1577.

[33] Isola, P., Zhu, J.-Y., Zhou, T., and Efros, A. A. (2017). Image-to-image translation with conditional adversarial networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1125–1134.

[34] Jain, V. and Learned-Miller, E. (2010). Fddb: A benchmark for face detection in unconstrained settings. Technical Report UM-CS-2010-009, University of Massachusetts, Amherst.

[35] Kalman, R. E. (1960). A new approach to linear filtering and prediction problems.

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

[37] Knapič, A., Malhi, A., Saluja, R., and Främling, K. (2021). xplainable artificial intelligence for human decision-support system in medical domain. *arXiv*.

[38] Kristan, M., Leonardis, A., Matas, J., Felsberg, M., Pflugfelder, R., Kämäräinen, J.-K., Danelljan, M., Zajc, L. Č., Lukežič, A., Drbohlav, O., et al. (2020). The eighth visual object tracking vot2020 challenge results. In *European Conference on Computer Vision*, pages 547–601. Springer.

[39] Kristan, M., Leonardis, A., Matas, J., Felsberg, M., Pflugfelder, R., ˇCehovin Zajc, L., Vojir, T., Hager, G., Lukezic, A., Eldesokey, A., et al. (2017). The visual object tracking vot2017 challenge results. In *Proceedings of the IEEE international conference on computer vision workshops*, pages 1949–1972.

[40] Kristan, M., Matas, J., Leonardis, A., Felsberg, M., Pflugfelder, R., Kämäräinen, J.-K., Chang, H. J., Danelljan, M., Cehovin, L., Lukežič, A., et al. (2021). The ninth visual object tracking vot2021 challenge results. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 2711–2738.

[41] Kristan, M., Matas, J., Leonardis, A., Vojíř, T., Pflugfelder, R., Fernandez, G., Nebehay, G., Porikli, F., and Čehovin, L. (2016a). A novel performance evaluation methodology for single-target trackers. *IEEE transactions on pattern analysis and machine intelligence*, 38(11):2137–2155.

[42] Kristan, M., Matas, J., Leonardis, A., Vojir, T., Pflugfelder, R., Fernandez, G., Nebehay, G., Porikli, F., and Čehovin, L. (2016b). A novel performance evaluation methodology for single-target trackers. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 38(11):2137–2155.

[43] Krizhevsky, A., Hinton, G., et al. (2009). Learning multiple layers of features from tiny images.

[44] LeCun, Y. and Cortes, C. (2010). MNIST handwritten digit database.

[45] Lewis, J. (1994). Fast template matching. *Vis. Interface*, 95.

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

[47] Liang, P., Blasch, E., and Ling, H. (2015). Encoding color information for visual tracking: Algorithms and benchmark. *IEEE transactions on image processing*, 24(12):5630–5644.

[48] Lin, T.-Y., Goyal, P., Girshick, R., He, K., and Dollár, P. (2017). Focal loss for dense object detection. In *Proceedings of the IEEE international conference on computer vision*, pages 2980–2988.

[49] Lin, T.-Y., Maire, M., Belongie, S., Bourdev, L., Girshick, R., Hays, J., Perona, P., Ramanan, D., Zitnick, C. L., and Dollár, P. (2014). Microsoft coco: Common objects in context.

[50] Liu, W., Anguelov, D., Erhan, D., Szegedy, C., Reed, S., Fu, C.-Y., and Berg, A. C. (2016). Ssd: Single shot multibox detector. In *European conference on computer vision*, pages 21–37. Springer.

[51] Loshchilov, I. and Hutter, F. (2017). Decoupled weight decay regularization. *arXiv preprint arXiv:1711.05101*.

[52] Lu, Y., Zhong, A., Li, Q., and Dong, B. (2018). Beyond finite layer neural networks: Bridging deep architectures and numerical differential equations. In *International Conference on Machine Learning*, pages 3276–3285. PMLR.

[53] McConnell, R. K. (1986). Method of and apparatus for pattern recognition. US Patent 4,567,610.

[54] Mencar, C. and Alonso, J. (2018). Paving the way to explainable artificial intelligence with fuzzy modeling. volume 24, pages 215–227.

[55] Mendel, J. M. and Bonissone, P. P. (2019). Critical thinking about explainable ai (xai) for rule-based fuzzy systems. *IEEE Transactions on Fuzzy Systems*, 14(1):69–81.

[56] Miller, G. A. (1998). *WordNet: An electronic lexical database*. MIT press.

[57] Mueller, M., Smith, N., and Ghanem, B. (2016). A benchmark and simulator for uav tracking. In *European conference on computer vision*, pages 445–461. Springer.

[58] Muller, M., Bibi, A., Giancola, S., Alsubaihi, S., and Ghanem, B. (2018). 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.

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

[60] O Pinheiro, P. O., Collobert, R., and Dollár, P. (2015). Learning to segment object candidates. *Advances in neural information processing systems*, 28.

[61] Pinheiro, P. O., Lin, T.-Y., Collobert, R., and Dollár, P. (2016). Learning to refine object segments. In *European conference on computer vision*, pages 75–91. Springer.

[62] Qi, Z., Khorram, S., and Li, F. (2019). Visualizing deep networks by optimizing with integrated gradients. In *CVPR Workshops*, volume 2.

[63] Qian, N. (1999). On the momentum term in gradient descent learning algorithms. *Neural networks*, 12(1):145–151.

[64] Qilong Wang, Banggu Wu, P. Z. P. L. W. Z. and Hu, Q. (2020). Eca-net: Efficient channel attention for deep convolutional neural networks. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*.

[65] Real, E., Shlens, J., Mazzocchi, S., Pan, X., and Vanhoucke, V. (2017). Youtube-boundingboxes: A large high-precision human-annotated data set for object detection in video. In *proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 5296–5305.

[66] Ren, S., He, K., Girshick, R., and Sun, J. (2015). Faster r-cnn: Towards real-time object detection with region proposal networks. *Advances in neural information processing systems*, 28.

[67] Rezatofighi, H., Tsoi, N., Gwak, J., Sadeghian, A., Reid, I., and Savarese, S. (2019). Generalized intersection over union: A metric and a loss for bounding box regression. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 658–666.

[68] Rifai, S., Vincent, P., Muller, X., Glorot, X., and Bengio, Y. (2011). Contractive auto-encoders: Explicit invariance during feature extraction. In *Icml*.

[69] Rocco, I., Arandjelovic, R., and Sivic, J. (2017). Convolutional neural network architecture for geometric matching. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 6148–6157.

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

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

[72] Russakovsky, O., Deng, J., Su, H., Krause, J., Satheesh, S., Ma, S., Huang, Z., Karpathy, A., Khosla, A., Bernstein, M., Berg, A. C., and Fei-Fei, L. (2015). ImageNet Large Scale Visual Recognition Challenge. *International Journal of Computer Vision (IJCV)*, 115(3):211–252.

[73] Scao, T. L. (2020). Neural differential equations for single image super-resolution. *arXiv preprint arXiv:2005.00865*.

[74] Schölkopf, B., Smola, A. J., Bach, F., et al. (2002). *Learning with kernels: support vector machines, regularization, optimization, and beyond*. MIT press.

[75] Selvaraju, R. R., Cogswell, M., Das, A., Vedantam, R., Parikh, D., and Batra, D. (2017). Grad-cam: Visual explanations from deep networks via gradient-based localization. In *Proceedings of the IEEE international conference on computer vision*, pages 618–626.

[76] Shrikumar, A., Greenside, P., and Kundaje, A. (2017). Learning important features through propagating activation differences. In *International conference on machine learning*, pages 3145–3153. PMLR.

[77] Sundararajan, M., Taly, A., and Yan, Q. (2017). Axiomatic attribution for deep networks. In *International conference on machine learning*, pages 3319–3328. PMLR.

[78] Tian, Z., Shen, C., Chen, H., and He, T. (2021). FCOS: A simple and strong anchor-free object detector.

[79] Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, Ł., and Polosukhin, I. (2017). Attention is all you need. *Advances in neural information processing systems*, 30.

[80] Viola, P. and Jones, M. (2001). Rapid object detection using a boosted cascade of simple features. In *Proceedings of the 2001 IEEE computer society conference on computer vision and pattern recognition. CVPR 2001*, volume 1, pages I–I. Ieee.

[81] Wang, Q., Zhang, L., Bertinetto, L., Hu, W., and Torr, P. H. (2019). Fast online object tracking and segmentation: A unifying approach. In *Proceedings of the IEEE/CVF conference on Computer Vision and Pattern Recognition*, pages 1328–1338.

[82] Wu, Y., Lim, J., and Yang, M.-H. (2013). Online object tracking: A benchmark. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 2411–2418.

[83] Wu, Y., Lim, J., and Yang, M.-H. (2015). Object tracking benchmark. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 37(9):1834–1848.

[84] Xu, K., Ba, J., Kiros, R., Cho, K., Courville, A., Salakhudinov, R., Zemel, R., and Bengio, Y. (2015). Show, attend and tell: Neural image caption generation with visual attention. In *International conference on machine learning*, pages 2048–2057. PMLR.

[85] Yan, B., Peng, H., Fu, J., Wang, D., and Lu, H. (2021a). Learning spatio-temporal transformer for visual tracking. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 10448–10457.

[86] Yan, B., Zhang, X., Wang, D., Lu, H., and Yang, X. (2021b). Alpha-refine: Boosting tracking performance by precise bounding box estimation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 5289–5298.

[87] Yang, Z., Yang, D., Dyer, C., He, X., Smola, A., and Hovy, E. (2016). Hierarchical attention networks for document classification. In *Proceedings of the 2016 conference of the North American chapter of the association for computational linguistics: human language technologies*, pages 1480–1489.

[88] Yu, C., Wang, J., Peng, C., Gao, C., Yu, G., and Sang, N. (2018). Learning a discriminative feature network for semantic segmentation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1857–1866.

[89] Yu, J., Jiang, Y., Wang, Z., Cao, Z., and Huang, T. (2016). Unitbox: An advanced object detection network. In *Proceedings of the 24th ACM international conference on Multimedia*, pages 516–520.

[90] Yu, Y., Xiong, Y., Huang, W., and Scott, M. R. (2020). Deformable siamese attention networks for visual object tracking. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 6728–6737.

[91] Yuan, K., Guo, S., Liu, Z., Zhou, A., Yu, F., and Wu, W. (2021). Incorporating convolution designs into visual transformers. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 579–588.

[92] Zhang, H., Goodfellow, I., Metaxas, D., and Odena, A. (2019). Self-attention generative adversarial networks. In *International conference on machine learning*, pages 7354–7363. PMLR.

[93] Zhang, Z., Peng, H., Fu, J., Li, B., and Hu, W. (2020). Ocean: Object-aware anchor-free tracking. In *European Conference on Computer Vision*, pages 771–787. Springer.

[94] Zhou, B., Khosla, A., Lapedriza, A., Oliva, A., and Torralba, A. (2016). Learning deep features for discriminative localization. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 2921–2929.