Segmenting and Tracking Objects in Video

Von der Fakultät für Mathematik, Informatik und Naturwissenschaften der RWTH Aachen University zur Erlangung des akademischen Grades eines Doktors der Naturwissenschaften genehmigte Dissertation

vorgelegt von

Diplom-Informatiker
Ali Athar
aus Karachi, Pakistan

Berichter: Prof. Dr. Bastian Leibe
Prof. Dr. Deva Ramanan

Tag der mündlichen Prüfung: 05.03.2024

Diese Dissertation ist auf den Internetseiten der Universitätsbibliothek online verfügbar.

Abstract

Research to develop methods that can accurately localize and track objects in video has been ongoing for decades. Approaches capable of accomplishing this are highly sought after for a variety of applications including autonomous robots, self-driving vehicles, sports analytics, video editing, etc. Despite significant progress in recent times, the task is far from solved, in particular for challenging scenarios involving occlusions, motion blur, and camera ego-motion. In this thesis, we present a series of works that advance the state of research in this domain in various ways, as outlined below.

Our first work, STEm-Seg, is an end-to-end trainable method for instance segmentation that models the input video as a single 3D space-time volume and relies on clustering per-pixel embeddings to segment and track objects. This differs from existing approaches, which largely follow the tracking-by-detection paradigm. Our novel formulation for these embeddings enables us to cluster the embeddings in an efficient and end-to-end learned fashion.

The second work, called HODOR, is aimed at mitigating the need for densely annotated data for training video tracking methods. Specifically, it tackles the task of Video Object Segmentation (VOS) in a weakly supervised manner where it can be trained using static images or sparsely annotated video. To this end, we adopt a novel approach that encodes objects into concise descriptors. This is in contrast to existing approaches that predominantly learn space-time correspondences, which makes it challenging to train them in such a setting.

Whereas the two aforementioned works propose network architectures, our third project proposes a dataset and benchmark called BURST that aims to unify the current, fragmented landscape of datasets in video segmentation research. BURST includes a benchmark suite that evaluates multiple tasks related to object segmentation in video with shared data and consistent evaluation metrics. The idea behind this is to facilitate knowl-

edge exchange between the research sub-communities tackling these tasks and also to encourage the development of methods with multi-task capability.

Finally, our fourth work, TarViS, can be seen as a logical continuation of the above in that it is a method that can tackle multiple video segmentation tasks. To achieve this, we decouple the task definition from the core network architecture and use a set of dynamic query inputs to specify the task-specific segmentation targets. This formulation enables us to train a single model jointly on a collection of datasets spanning multiple tasks (Video Instance/Object/Panoptic Segmentation). During inference, the model can switch between tasks by simply hot-swapping the input queries accordingly.

Zusammenfassung

Die genaue Lokalisierung und Verfolgung von Objekten in Videos ist seit Jahrzehnten Thema aktiver Forschung. Entsprechende Methoden sind für eine Vielzahl von Anwendungen höchst relevant, darunter autonome Roboter, selbstfahrende Fahrzeuge, Sportanalysen und Videobearbeitung. Trotz erheblicher Fortschritte in jüngster Zeit ist die Aufgabe noch lange nicht gelöst, insbesondere bei schwierigen Szenarien mit Verdeckungen, Bewegungsunschärfe und Eigenbewegung der Kamera. Diese Dissertation umfasst vier unserer Arbeiten, die den Stand der Forschung in diesem Bereich vorangetrieben haben, wie im Folgenden erläutert wird.

Die erste Methode, STEm-Seg, ist eine ganzheitlich trainierbare Architektur für Video Instance Segmentation, die das Eingabevideo als 3D-Raum-Zeit-Volumen modelliert. Objektsegmentierung und -verfolgung werden dabei durch Clustering von Pixel Embeddings realisiert, im Unterschied zu bestehenden Ansätzen, die weitgehend dem Paradigma des Tracking-by-Detection folgen. Unsere neuartige Formulierung ermöglichte es uns, die Embeddings in einer effizienten und durchgängig erlernten Weise zu clustern.

Unsere zweite Arbeit mit dem Namen HODOR zielt darauf ab, den Bedarf an vollständig annotierten Daten für das Training von Video-Tracking-Methoden zu verringern. Insbesondere ermöglicht unser neuartiger Ansatz für Video Object Segmentation das Training auf statischen Bildern oder einzelnen Videoframes. HODOR beschreibt Objekte mit präzisen Deskriptoren, die eine robuste Verarbeitung auch mit wenigen Trainingsdaten erlauben. Bestehende Ansätze lernen größtenteils Raum-Zeit-Korrespondenzen, die mit solchen Daten nur schwer zu trainieren sind.

Während sich die beiden vorgenannten Arbeiten mit Netzwerkarchitekturen beschäftigen, schlägt unser drittes Projekt einen Datensatz und Benchmark namens BURST vor, der die derzeitige, fragmentierte Landschaft von Datensätzen in der Videosegmentierungsforschung vereinheitlichen soll. BURST umfasst eine Benchmark-Suite, die mehrere

ähnliche Aufgaben im Zusammenhang mit der Objektsegmentierung in Videos auf gemeinsamen Daten und mit konsistenten Metriken zur Bewertung evaluiert. Die Idee dahinter ist, den Wissensaustausch zwischen den Gruppen, die sich mit Videosegmentierung beschäftigen, zu erleichtern und die Entwicklung von Ansätzen mit Multitasking-Fähigkeit zu fördern.

Unsere vierte Arbeit, TarViS, verfolgt ebenso das Ziel der Vereinigung unterschiedlicher Formen der Videosegmentierung. Um dies zu erreichen, entkoppeln wir die Aufgabendefinition von der Netzwerkarchitektur und spezifizieren die jeweiligen Segmentierungsziele mit dynamischen Queries. Diese Formulierung ermöglicht es uns, ein einziges Modell auf einer Sammlung von Datensätzen zu trainieren, die mehrere verschiedene Aufgaben umfassen (Videoinstanz/Objekt/panoptische Segmentierung). Während der Inferenz kann mit den entsprechenden Queries dynamisch zwischen allen Aufgaben gewechselt werden.

Acknowledgments

Although this is 'my' doctoral thesis, its completion would not have been possible without the collaboration, mentorship, and support of several individuals to whom I would like to extend my heartfelt gratitude.

To begin with, I thank my supervisor, Prof. Dr. Bastian Leibe, for allowing me to be a Ph.D. candidate in his group and for his invaluable mentorship, feedback, and guidance throughout these years. Additionally, I would like to thank Prof. Dr. Deva Ramanan for co-examining this thesis, and for his collaboration and feedback on multiple projects that are also part of this thesis.

The research presented in this thesis would not have been possible without the efforts of my colleagues at the RWTH Vision Lab with whom I frequently collaborated and also consider to be good friends: Alexander Hermans, Jonathon Luiten, Sabarinath Mahadevan, Aljosa Osep, and Paul Voigtlaender. I would also like to thank all the students I worked with over the years for placing their trust in my supervision capability: Christian Schmidt, Sebastian Hennen, Aleksander Jovanov, and Blin Beqa. Moreover, I would like to thank Enxu Li and Sergio Casas for their collaborative effort and supervision during my internship at Waabi.

Aside from work-related interactions, I consider myself lucky to have spent the last five-odd years in the company of individuals who are not only great colleagues but also pleasant people to socialize and be friends with. Over the years I have spent many weekends socializing with them late into the night, going bouldering (shout out to Alexey Nekrasov), and having fun-filled travel experiences at overseas conferences. I will always cherish these fond memories and I hope our paths cross again in the future.

On a more personal note, I would like to thank my parents and sister for their unconditional support and for giving me a shoulder on which I could always lean for help

and counsel. Lastly, being a practicing Muslim, I thank God for His countless blessings which have enabled me to come this far in life, and which I pray will continue in the future.

Contents

1 Introduction **1**
- 1.1 Motivation . . . 1
- 1.2 Contributions . . . 3
- 1.3 Structure of the Thesis . . . 5

2 State of the Art **9**
- 2.1 Classical Approaches . . . 9
- 2.2 Early Deep Learning-based Approaches . . . 12
- 2.3 Recent Attention- and Transformer-based Approaches . . . 14
- 2.4 Datasets and Benchmarks . . . 17

3 Preliminaries **19**
- 3.1 ResNet . . . 19
- 3.2 Multi-head Attention . . . 24
- 3.3 Transformer Networks . . . 27
- 3.4 Evaluation Measures . . . 38

4 STEm-Seg: Spatio-temporal Embeddings for Object Segmentation in Video **43**
- 4.1 Introduction . . . 43
- 4.2 Related Work . . . 45
- 4.3 Our Method . . . 47
- 4.4 Experimental Evaluation . . . 53
- 4.5 Conclusion . . . 59

5 HODOR: High-level Object Descriptors for Object Re-segmentation in Video Learned from Static Images **61**
- 5.1 Introduction . . . 61

5.2 Related Work . . . 64
5.3 Method . . . 65
5.4 Experimental Evaluation . . . 70
5.5 Limitations . . . 77
5.6 Conclusion . . . 77

6 BURST: A Benchmark for Unifying Object Recognition, Segmentation, and Tracking in Video 79
6.1 Introduction . . . 79
6.2 The BURST Benchmark . . . 82
6.3 Comparison to Related Datasets . . . 83
6.4 Dataset Creation . . . 84
6.5 BURST Task Taxonomy . . . 86
6.6 Unified Evaluation Metrics . . . 88
6.7 Baselines . . . 90
6.8 Conclusion . . . 94

7 TarViS: A Unified Approach for Target-based Video Segmentation 99
7.1 Introduction . . . 99
7.2 Related Work . . . 101
7.3 Method . . . 103
7.4 Experiments . . . 109
7.5 Discussion . . . 113
7.6 Query Visualization . . . 114
7.7 Conclusion . . . 117

8 Conclusion 119
8.1 Perspectives . . . 120

Bibliography 123

1 Introduction

1.1 Motivation

Video understanding is a fundamental research area in computer vision. In general, it involves developing algorithms and techniques that can extract meaningful information from video data. This encompasses a wide array of tasks which include classifying and recognizing objects, events, and activities, and also comprehending the complex relationships between them by leveraging the content and temporal order of the video frames.

This thesis focuses on a branch of video understanding that involves pixel-precise segmentation of objects and semantic classes. In computer vision parlance, such problems are sometimes referred to as "dense prediction" or "video segmentation" tasks since the focus is on precisely localizing and tracking entities (e.g. objects) across the input video sequence. The ability to effectively perform such tasks is crucial for video editing, sports analysis, and more importantly, building perception systems for self-driving vehicles, autonomous robots, and surveillance applications.

1.1.1 Why Images?

At this point, one could argue that from a practical perspective, utilizing 3D data in the form of point-clouds captured from LiDAR sensors would yield better scene understanding results. While it is true that 3D data contains more spatial context about the scene, there are often size and cost constraints when building perception systems, and for such cases being able to work with camera video is desirable since camera sensors are much more cost-effective and far more ubiquitous than 3D-capable LiDAR sensors. Furthermore, the amount of publicly available video data (both labeled and unlabeled) far outstrips the amount of available point-cloud data.

Moreover, from a research perspective, if we are pursuing the lofty goal of developing 'AI systems' that can function as well as humans, then it should be possible to accurately localize objects in a scene from 2D images since humans can easily achieve the same even if we disregard our stereo vision ability (in several countries, one can legally drive a car even if one eye is completely blind).

1.1.2 Constituent Tasks

As is the case with most research areas in computer vision, there are a number of standardized datasets and benchmarks for evaluating video segmentation performance. Although these benchmarks differ in terms of the exact task definition and the type of video data used for evaluation, they share a common theme: segmenting and tracking objects and/or semantic classes across video sequences. A few popular tasks which we will tackle in the upcoming chapters of this thesis are listed below:

- **Video Instance Segmentation (VIS):** This task involves segmenting, tracking and classifying all objects in a video that belong to a set of pre-defined object classes e.g. *car*, *person*, *cat*, etc.. Popular datasets for VIS include YouTube-VIS (Yang *et al.*, 2019a) and OVIS (Qi *et al.*, 2022). The motivation behind these benchmarks was to extend the task of image-level instance segmentation to the video domain. The datasets for this task typically contain general, in-the-wild videos from the internet.

- **Multi-object Tracking and Segmentation (MOTS):** This benchmark was introduced at about the same time as VIS, and the task definition is identical: given a video, all objects belonging to a set of pre-defined classes have to be segmented and assigned temporally consistent IDs and class labels. However, VIS and MOTS have different motivations and origins: the latter is built upon the Multi-object Tracking (MOT) task which requires box-level tracking of objects. MOTS extends this and requires pixel-precise segmentation masks instead of coarse bounding boxes. Benchmarks for this task (Voigtlaender *et al.*, 2019b) are more focused on driving and street scenes, and the set of object classes is smaller than that for VIS.

- **Video Object Segmentation (VOS):** This name may sound very similar to VIS, but the task definition here is different since it requires segmenting and tracking a set of arbitrary objects for which the ground-truth masks are provided for the first video frame in which each object occurs. Thus, unlike VIS and MOTS, VOS is a class-agnostic task since methods are required to segment and track objects belonging to arbitrary categories. Currently, the two most popular VOS datasets are DAVIS (Pont-Tuset *et al.*, 2017) and YouTube-VOS (Xu *et al.*, 2018b).

- **Video Panoptic Segmentation (VPS):** This task is an extension of the image-level panoptic segmentation task (Kirillov *et al.*, 2019) to video. It can be seen as an extension of VIS and MOTS where, in addition to segmenting and tracking objects belonging to a set of pre-defined object classes, methods are also required to segment non-instantiable *stuff* classes e.g. *sky*, *grass*, *wall*. In practice, VPS benchmarks require every pixel in the video to be assigned to some object instance or stuff class. Popular datasets for VPS include Cityscapes-VPS (Kim *et al.*, 2020) and KITTI-STEP (Weber *et al.*, 2021) which focus on driving scenes, and the recently introduced VIPSeg dataset (Miao *et al.*, 2022) which is larger and contains in-the-wild videos from the internet.

Illustrated examples of the above-mentioned tasks are given in Fig. 1.1.

1.2 Contributions

In this thesis, we present multiple research works pertaining to the tasks listed in Sec. 1.1.2. Three of these projects propose novel network architectures that improve the state-of-the-art on one or more tasks, whereas one project proposes a new dataset and associated benchmark for object segmentation in video.

Bottom-up, End-to-End Architecture for VIS and MOTS: In Chapter 4, we propose a novel approach for Video Instance Segmentation (VIS), Multi-object Tracking and Segmentation (MOTS), and so-called 'Unsupervised' Video Object Segmentation (UVOS). At the time when this project was undertaken, methods tackling these tasks were largely based on the tracking-by-detection paradigm which involves first detecting/segmenting objects in individual image frames, followed by a temporal association step to yield object tracks that are consistent over time. Our proposed architecture, called STEm-Seg, proposes an alternate approach that models a video sequence as a single 3D spatio-temporal volume and jointly segments objects across space and time in a single step. To achieve this, STEm-Seg learns per-pixel embeddings and clustering parameters for the input scene, which allows us to efficiently segment objects across the entire input video. We apply our approach to three popular benchmarks and achieve state-of-the-art results on all three.

Weakly Supervised VOS With Static Images and Sparsely Annotated Video: The contribution of the work presented in Chapter 5 is to circumvent a fundamental limitation of video segmentation methods, namely, the fact that they need large amounts of temporally dense and pixel-precise mask annotations for training. To overcome this, we propose a weakly supervised method for Video Object Segmentation called HODOR which can be trained on labeled static images, but during inference is able to track objects in video. The motivation for this comes from the fact that static images require

much less annotation effort compared to video and that there are already large-scale, publicly available datasets containing hundreds of thousands of annotated images. To achieve this, we propose an encoder-decoder architecture for HODOR where the encoder learns to summarize a given object as a concise descriptor which can then be decoded into another video frame to segment the object there. We apply HODOR to multiple VOS benchmarks and show that it achieves state-of-the-art performance among methods that only utilize image-level supervision.

Universal Video Segmentation: In Chapters 6 and 7 of this thesis, we explore the possibility of developing generalized video segmentation approaches with multi-task capability. This differs starkly from the current research landscape where the focus of most works is on tackling one of the multiple tasks discussed in Sec. 1.1.2. To this end, our contribution is two-fold, as stated below:

- In Chapter 6, we contribute a large-scale public dataset called BURST with pixel-precise mask annotations for several thousand videos with diverse scenes, and an associated benchmark containing six different tasks. BURST is based on the existing TAO dataset (Dave *et al.*, 2020) and contains a large set of annotated object classes. The motivation for creating this dataset comes from the fact that video segmentation is currently fractured into a number of different benchmarks and tasks, each with its own sub-community of researchers, even though these tasks (e.g. those listed in Sec. 1.1.2) are conceptually similar. The idea behind BURST is to unify these disparate benchmarks under a single umbrella with shared data and evaluation metrics.
- Having published a dataset and benchmark for evaluating multiple video segmentation tasks, the aim of our next project (Chapter 7) was to develop a unified method that can be applied to these tasks. To achieve this, we propose a network architecture called TarViS which tackles the general task of segmenting a set of arbitrary 'targets' from a given input video. The underlying idea is that the tasks listed in Sec. 1.1.2 can all be seen as specializations of this task since they differ only in terms of how the targets are defined. To implement this concept, we decouple the task definition from the core network architecture by encoding the segmentation targets as a set of concise 'query' features which are input to a transformer-based architecture together with the video features. The target queries are subsequently refined and used to predict segmentation masks that are supervised with the corresponding ground-truth. This way, we are able to train TarViS with a collection of datasets spanning multiple tasks, and during inference, we can switch between tasks by simply hot-swapping the query inputs to the network. We apply our approach to four different tasks (VIS, VOS, VPS, PET) and show that it achieves state-of-the-art performance on five out of the seven benchmarks that we evaluate.

1.3 Structure of the Thesis

The remainder of this thesis is structured as follows: Chapters 2 and 3 discuss literature and preliminary concepts relevant to this thesis, respectively. Then, Chapters 4-7 discuss each of the four projects discussed above. Finally, Chapter 8 offers concluding remarks. The content of each of the chapters is outlined in the following paragraphs.

Chapter 2: State-of-the-art. Here we provide an overview of the evolution of video segmentation research from early works in $\sim$ 2010 to the state-of-the-art approaches we see today. We will explain the genesis of this research domain with early works that focused on low-level concepts such as motion segmentation and template-based tracking. Then, we will cover how research evolved to high-level concepts such as objects and semantic classes, and how the state-of-the-art for such tasks has dramatically improved since the advent of the deep learning era. Finally, we will categorize and discuss recent approaches and benchmarks for video segmentation with a particular focus on literature relevant to the tasks in Sec. 1.1.2.

Chapter 3: Preliminaries. In this chapter, we will go over fundamental concepts that are repeatedly used in the remaining chapters. This includes a discussion of popular backbone networks used for learning image features e.g. ResNet (He *et al.*, 2016) and Swin-Transformer (Liu *et al.*, 2021), architecture building blocks such as Transformer networks, Multi-head Attention (Vaswani *et al.*, 2017), Deformable Attention (Zhu *et al.*, 2020) and 3D convolutions for video understanding. Moreover, we will explain the metrics and evaluation measures that are employed for video segmentation tasks and discuss the merits and demerits of each of them.

Chapter 4: STEm-Seg: Spatio-temporal Embeddings for Object Segmentation in Video. This is the first of four projects where we develop a method for tracking and segmenting objects in video using a clustering-based approach that operates on a 3D spatio-temporal volume, thereby segmenting objects over space and time in a unified manner. STEm-Seg works by producing per-pixel embeddings and also the parameters needed to cluster embeddings belonging to the same object, thereby enabling object clusters to be inferred in an efficient and end-to-end learned fashion. This chapter is based on research that was originally published at the European Conference on Computer Vision (ECCV) (Athar *et al.*, 2020) along with additional benchmark results presented at the "Occluded Video Instance Segmentation" workshop at the Conference on Computer Vision and Pattern Recognition (CVPR) (Athar *et al.*, 2021).

Chapter 5: HODOR: High-level Object Descriptors for Object Re-segmentation in Video Learned from Static Images. This is the second of four projects where we develop a weakly-supervised method for Video Object Segmentation (VOS) which utilizes labeled, static images for training, but can be applied to temporal video sequences during inference. This was achieved by employing an encoder-decoder architecture that

can encode objects into robust and concise 'descriptors' which are then decoded into other video frames to 're-segment' the objects there. Our end-to-end trainable workflow also enables us to use unlabeled video frames for training by employing cyclic consistency for supervision. This chapter is based on research that was originally published at the Conference on Computer Vision and Pattern Recognition (CVPR) (Athar *et al.*, 2022).

Chapter 6: BURST: Benchmark for Unifying Object Recognition, Segmentation and Tracking in Video. This is the third of four projects where we publish a new public dataset and associated benchmark. The idea behind this project is to unify multiple tasks related to object segmentation and tracking in video under a single umbrella with shared data and evaluation measures. This is desirable since the research community is currently fragmented across multiple datasets and benchmarks with different evaluation measures even though the tasks which they tackle share key commonalities. Moreover, the BURST dataset comprises video sequences that are lengthier and more diverse in terms of the video scene, and the number of object classes that are annotated. This chapter is based on research that was originally published at the Winter Conference on Computer Vision (WACV) (Athar *et al.*, 2023a).

Chapter 7: TarViS: A Unified Approach for Target-based Video Segmentation. This is the final (fourth) project where we propose a method that is able to tackle multiple tasks involving dense prediction in video. The underlying idea is that all such tasks are instantiations of a generic, abstract task that requires segmenting a set of 'targets' in video. The instantiated tasks differ only in terms of how the targets are defined. To realize this concept, we propose a network architecture wherein the targets are encoded as 'queries' that are input, together with the video frames, to a transformer-based network that predicts segmentation masks for each of the input queries. The key idea is that the network is agnostic to how the targets are defined, thus decoupling the task definition from the architecture. This enables us to train on a collection of datasets spanning multiple tasks, and during inference, our model can switch between tasks by simply hot-swapping the input queries without requiring any task-specific retraining. This chapter is based on research that was originally published at the Conference on Computer Vision and Pattern Recognition (CVPR) (Athar *et al.*, 2023b).



Figure 1.1: Visualization of the inputs and outputs for various video segmentation tasks.

2 State of the Art

In this section, we provide a holistic overview of existing literature pertaining to video segmentation and object tracking in video. The first three sections can be divided chronologically: Sec. 2.1 broadly covers classical approaches used for motion and video segmentation and object tracking up until $\sim$ 2015 i.e. prior to the emergence of deep learning-based methods. Subsequently, Sec. 2.1 covers approaches that began employing deep learning and CNN architectures; the time frame for these works roughly spans from 2016 to 2019. Thereafter, in Sec. 2.3 we discuss more recent approaches (2020 and onward) that emerged after the popularization of transformer-based architectures involving multi-head attention in computer vision. Finally, in Sec. 2.4 we will describe the dataset and benchmark landscape in this domain and how it evolved over time.

2.1 Classical Approaches

Even prior to the deep learning boom, segmenting patterns and tracking objects in video was a popular research domain. Several early approaches were based on the Perceptual Grouping Theory and Gestalt principles according to which humans perceive and associate visual elements based on their proximity, appearance similarity, continuity, and closure. The family of approaches that adopted this paradigm to tackle video segmentation adopted a bottom-up approach to group pixels across space (within frames) and time (across frames). Brox and Malik (2010) tackled object segmentation by first generating point-level trajectories using an optical flow-based tracker (Sundaram *et al.*, 2010) and then performed spectral clustering guided by the similarities between these trajectories to group pixels into object instances. Related to this, Ochs and Brox (2012) tackled motion segmentation in video using point trajectories. However, instead of only utilizing pairwise similarities between trajectories, they performed higher-order comparisons between groups of trajectories to better model rotation and scaling motion.

Another set of bottom-up approaches tackled the problem by extending the image-level concept of superpixels to video by grouping pixels that share similar appearance and evolve in a similar fashion over time. Xu (2012) referred to these as 'super-voxels' because of their 3D nature and experimented with five different super-voxel methods based on weighted aggregation, graph-based, hierarchical graph-based, mean-shift based and Nyström normalized cuts, respectively. These methods were applied to three different datasets which mostly involved annotations of humans. Meanwhile, Chang *et al.* (2013) proposed 'temporal superpixels' which model inter-frame motion using a Gaussian process with optical flow guidance.

2.1.1 Box-level Multi-object Tracking

In parallel to research on low-level motion-based segmentation, there was also a large family of methods tackling bounding box-level tracking of multiple objects (MOT) in video. These approaches were generally based on the tracking-by-detection paradigm where object detections are first predicted for individual image frames, followed by an inter-frame association step to generate temporally consistent object tracks. Before the emergence of convolutional networks for localization tasks, object detection was commonly tackled by using selective search (Uijlings *et al.*, 2013), deformable parts model (Felzenszwalb *et al.*, 2009), implicit shape models (Leibe *et al.*, 2008b) and sliding window approaches involving Histograms of Oriented Gradients (HOG) (Dalal and Triggs, 2005).

The task of associating per-frame detections across time can be seen as an optimization problem. The focus of MOT approaches at the time was on formulating this problem in an efficient way that could reason over the appearance of the objects as well as the complex and non-linear nature of object movement over time. Pirsiavash *et al.* (2011) formulated the association as a min-cost network flow problem and proposed an efficient, sequential greedy algorithm to obtain near-optimal solutions. Dicle *et al.* (2013) initialized short tracklets based on the distances between detections, and then iteratively merged tracklets across increasing frame gaps by solving a linear assignment problem. To this end, a linear regression model was used to model tracklet motion.

Meanwhile, Milan *et al.* (2013) formulated temporal association as an energy minimization problem where the energy function encoded appearance with HOG and Histogram of relative Optical Flow (HOF) features whereas the dynamics model was designed to incorporate track persistence, mutual exclusion and regularization to prevent too many object tracks. The energy function was minimized iteratively using conjugate gradient descent. Rezatofighi *et al.* (2015) formulate object track building as a Joint Probabilistic Data Association (JPDA) problem which they optimize by solving a series of Integer Linear Programs (ILP). Choi (2015) proposed an online approach that associates detections in the current frame with existing object tracks within a given temporal

history. To this end, they build an aggregated local flow descriptor that parameterizes detections by aggregating the point trajectories inside them. The actual association task is formulated as a multi-hypothesis problem which is optimized using Conditional Random Fields (CRF).

Looking at the aforementioned works, we note that their focus was on devising accurate similarity measures between object detections/tracks and formulating the association problem in a near-optimal way that was also computationally feasible.

2.1.2 Video Object Segmentation

Alongside research on box-level tracking, the task of Video Object Segmentation (VOS) was also being tackled in a variety of ways. Note that while mask-level segmentation of multiple objects (i.e. the VIS and MOTS tasks) was still very challenging to address at the time, VOS was still tractable because the task is simplified by the presence of first-frame ground-truth masks during inference. Early methods, similar to their modern counterparts, adopted a 'mask propagation' approach wherein the first-frame masks are transferred to subsequent video frames based on the affinities between image regions across time.

Grundmann *et al.* (2010) adopted a hierarchical approach that first over-segments a video volume based on color differences between adjacent pixels. This initial segmentation, referred to as a 'region graph', is then iteratively improved by merging similar segments based on their respective LAB color histograms and the optical flow between frames. Avinash Ramakanth and Venkatesh Babu (2014) proposed using 'temporal seams' to connect image-level patches across time using an energy minimization formulation where the energy function uses appearance features based on color and color gradients. Fan *et al.* (2015) proposed a method called 'JumpCut' which computes two separate 'nearest-neighbor field splits' to map the foreground and background mask regions from a given labeled frame to a target frame. This was followed by a level set-based approach to refine the masks by snapping to silhouette edges and smoothing the mask contours. Perazzi *et al.* (2015) proposed an approach that first generates a large number of object proposals for each image frame which are then pruned using a Support Vector Machine (SVM) based on their similarity to the given first-frame mask. The similarity is parameterized by color, position, area, HOG, and motion features from the Histogram of Oriented Optical Flow (HOOF). Finally, the pruned object proposals are assigned binary labels to decide whether they belong to the target object by performing MAP inference on a CRF where the unary potentials encode the proposal confidence and pairwise potentials encode color similarity, spatial proximity, and temporal consistency.

2.2 Early Deep Learning-based Approaches

The emergence of deep learning in computer vision fundamentally altered the course of research in video segmentation and object tracking. In the following subsections, we discuss how researchers began to leverage deep learning for such tasks.

2.2.1 Box-level Multi-object Tracking

Approaches tackling MOT were impacted mainly in two ways: (1) The quality of per-frame object bounding boxes improved significantly with the introduction of CNN-based object detectors such as YOLO (Redmon *et al.*, 2016) and Faster-RCNN (Ren *et al.*, 2015). (2) The need for complex, hand-crafted features for modeling object appearance and motion dynamics was mitigated by deep neural networks which could better learn these features from annotated data.

For instance, Tang *et al.* (2017) used a Siamese CNN architecture for computing pairwise object detection appearance similarities, which were then concatenated with spatio-temporal proximity and deep learning-based keypoint features to compute edge weights for a graph. Subsequently, the object detections (graph vertices) were clustered into tracks by formulating this as a Minimum Cost Lifted Multicut problem. Meanwhile, Sadeghian *et al.* (2017) proposed an online approach that uses separate, learned models for encoding appearance similarity, motion consistency, and inter-object interactions, respectively. These learned similarity cues were then input to an RNN which was trained to associate object detections in a given frame with the set of existing object tracks. On the other hand, Yoon *et al.* (2018) adapted a single object tracker for multi-object tracking by proposing spatial and temporal attention networks to improve association quality in challenging scenarios involving objects with similar appearances, occlusions, and noisy object detections.

Comparing the aforementioned approaches to the classical MOT approaches discussed in Sec. 2.1.1, we see a major paradigm shift. The pre-deep learning era approaches had to work with low-quality object detections and hand-crafted similarity features which were often unreliable. Thus, the burden of producing accurate object tracks was largely put on the subsequent association step which had to be carefully formulated to be computationally tractable, while also being as expressive and as optimal as possible. By contrast, the research focus in later works shifted towards designing network architectures for learning accurate similarity measures between object detections/tracks. The availability of these accurate measures, coupled with high-quality object detections from CNN architectures, enabled the subsequent association step to be tackled with simpler techniques such as linear assignment.

2.2.2 Video Object Segmentation

Similar to MOT, approaches tackling VOS also underwent significant changes due to the emergence of CNN architectures. We broadly categorize early deep learning-based VOS approaches into groups. The first group of methods finetune a CNN model at runtime in order to enable it to propagate a given object mask across a given video sequence. Caelles *et al.* (2017) employed an ImageNet pre-trained VGGNet (Simonyan and Zisserman, 2015) as a backbone for learning image features, followed by upsampling layers to regress a binary segmentation mask from the multi-scale backbone features. Their model was trained for generic foreground/background segmentation on video data, followed by inference-time finetuning for each object using the given first-frame ground-truth mask. Voigtlaender and Leibe (2017) improved the temporal robustness of this approach by additionally fine-tuning the model on its own predicted output for previous frames. Later, Khoreva *et al.* (2019) proposed an elaborate augmentation strategy involving mask cropping and image-inpainting which allowed for the generation of thousands of diverse training samples from the given first-frame ground-truth. Moreover, the input to their network included the predicted mask for the previous frame and optical flow as additional guidance cues.

Although these methods considerably improved performance over the earlier classical methods based on superpixels, point trajectories, and image patch associations, they suffered from slow runtimes because of the online finetuning step. The second group of approaches thus aimed to mitigate this shortcoming by proposing models that only had to be trained once and could then generalize to different objects and video sequences at runtime. Wug Oh *et al.* (2018) proposed a Siamese architecture to regress the mask for an object in the target frame where one branch accepts the reference frame image and mask as input whereas the second branch accepts the target frame and previous frame mask as input. Hu *et al.* (2018) adopted a conceptually similar approach where pixels in the current frame are assigned a binary label based on the learned feature similarity between them and the object/non-object pixels in the annotated first frame. Meanwhile, Chen *et al.* (2018) adopted a metric learning approach; their model produced per-pixel embeddings which formed tight clusters in the embedding space for a given object. During inference, the embeddings for the object/non-object pixels in the annotated first frame were used to assign labels to pixels in subsequent video frames based on the nearest-neighbor embedding distance. The work of Voigtlaender *et al.* (2019a) involved two-tier feature matching between the current and first frame, and between the current and previous frame, respectively. Their work offered significant runtime improvements over existing methods since most of the forward pass only had to be executed once for each video; only a lightweight 'dynamic segmentation head' had to be run individually for each object to predict its respective mask.

Whereas the above methods mostly relied on pixel-level mask propagation across frames, some works also adopted a more tracking-centric approach to VOS. For instance,

Cheng *et al.* (2018) decomposed the first frame object mask into multiple parts described by bounding boxes. These box-level parts were then propagated to future frames using tracking methods, followed by a regression step to obtain pixel-precise masks from the box-level parts. Meanwhile, Luiten *et al.* (2018) adapted the tracking-by-detection paradigm for VOS: they generated pixel-precise object proposals for the video frames and assigned these proposals to the first-frame objects by using optical flow and ReID embedding similarity as guidance.

2.2.3 Pixel-precise Multi-object Tracking

In the previous two sub-sections, we discussed how deep learning architectures enabled significant performance improvements in box-level MOT and VOS approaches. These developments also led to the creation of new tasks that were erstwhile infeasible to tackle with classical approaches. One such task that quickly gained traction in the community is that of pixel-precise segmentation and tracking of multiple tasks (i.e. the VIS and MOTS tasks explained in Sec. 1.1.2). Early approaches for this task can be seen as extensions of Mask-RCNN (He *et al.*, 2017), a widely popular image-level instance segmentation network.

To this end, Voigtlaender *et al.* (2019b) proposed Track-RCNN, which augments the Mask-RCNN architecture by adding another output head that produces an embedding vector for each object prediction. Per-frame detections are then associated across time by using the Euclidean distance between their respective embeddings as a similarity measure. Yang *et al.* (2019a) adopted a similar approach, but they also included the mask IoU and object class prediction into the similarity measure calculation. On the other hand, Bertasius and Torresani (2020) proposed adding a 'MaskProp' module to the Mask-RCNN architecture which propagates the instance mask in a given frame to other frames in the video clip by applying deformable convolutions and attention operations to masked features.

2.3 Recent Attention- and Transformer-based Approaches

In this section, we discuss recent works that emerged after the popularization of multi-head attention and transformer-based architectures (Vaswani *et al.*, 2017) in computer vision. This development spurred a paradigm shift in video segmentation and tracking approaches, and in particular for tasks requiring pixel-precise multi-object tracking. The seminal work of Carion *et al.* (2020) (commonly abbreviated as DETR) tackled image-level object detection by initializing a set of 'queries' which were trained to represent object instances in the input image by iteratively self-attending to each other, and cross-

attending to the image features in a round-robin fashion. This approach resulted in an end-to-end trainable object detector that required neither initial anchor boxes nor post-processing steps such as Non-Maximum Suppression (NMS). Later, Cheng *et al.* (2022) proposed Mask2Former, which extended this approach to tackle multiple tasks involving pixel-precise image segmentation.

Although box-level multi-object tracking is still an active research domain, in this section we will only discuss approaches pertaining to pixel-precise segmentation since these are more relevant for subsequent chapters of this thesis.

2.3.1 Pixel-precise Multi-object Tracking

Most recent state-of-the-art approaches tackling Video Instance Segmentation (VIS) and Multi-object Tracking and Segmentation (MOTS) are strongly inspired by the image-level DETR architecture (Carion *et al.*, 2020) in the sense that they utilize a set of 'queries' which are trained to capture the object instances in the input video. Wang *et al.* (2021c) proposed VisTr and were one of the first to apply this approach to VIS. VisTr employs a separate set of queries for each video frame where the i-th query for frame t is trained to segment the same object instance as the i-th query in frame $t + N$, thereby yielding temporally consistent object masks.

Meanwhile, other methods employ a combination of frame-level and video-level queries. This is intuitively appealing because it allows video segmentation approaches to readily utilize existing state-of-the-art image-level architectures for per-frame segmentation. Alternatively, this can be seen as a manifestation of the tracking-by-detection paradigm where frame-level queries segment objects in individual frames, followed by a second step in which video-level queries aggregate information over time. However, a major difference to the tracking-by-detection approaches discussed earlier (Sec. 2.1.1,2.2.3) is that these recent approaches are generally end-to-end trainable and do not require solving a graph-level optimization problem for temporal association.

Wu *et al.* (2022b) proposed SeqFormer, which decomposes a set of video-level queries to frame-level queries when applying cross-attention to image features. These frame-level queries are then collected and input to another attention-based decoder which transfers their information back to the video-level queries. Hwang *et al.* (2021) adopted a conceptually similar strategy by employing a two-step process where a set of memory tokens first learn per-frame features, followed by a second step in which the memory tokens for all frames are collected and applied to several attention layers to incorporate temporal context. Heo *et al.* (2022) proposed a model called VITA which learns frame-level queries using an architecture similar to Mask2Former (Cheng *et al.*, 2022) followed by a temporal aggregation step where a set of video-level queries cross-attend to the per-frame queries across the entire input clip.

On the other hand, some works adopt approaches inspired by metric learning where the image-level object queries are used as ReID vectors for temporal association. Huang *et al.* (2022) showed that object queries from a trained Mask2Former (Cheng *et al.*, 2022) model can be used to associate the per-frame predictions over time by computing their cosine similarities and performing Hungarian Matching. Wu *et al.* (2022a) proposed an online approach that is similar, but they additionally apply a contrastive learning loss to the object queries which enforces that queries belonging to the same object instance are similar across time and those belonging to different objects are dissimilar.

Although DETR-based approaches such as those discussed above are now highly popular, there are still some recent works that are conceptually similar to the early deep-learning approaches. These works are inspired by single-stage image-level detectors such as YOLO (Redmon *et al.*, 2016) and SOLO (Wang *et al.*, 2021b). Yang *et al.* (2021a) proposed CrossVIS, which performs per-frame segmentation by predicting dynamic convolutional kernels which are then applied to the frame features. To perform temporal association, CrossVIS predicts discriminative embeddings for each object and also enforces the dynamic kernels to be temporally consistent. Meanwhile, Han *et al.* (2022) propose VISOLO, where the image-level SOLO architecture (Wang *et al.*, 2021b) is extended to video by adding additional network components that enable feature interaction across time.

2.3.2 Video Object Segmentation

Recent approaches for VOS are largely based on the seminal work of Oh *et al.* (2019). Their method, abbreviated as STM-VOS, adopts an attention-based mechanism that involves computing affinities between the pixel features in the current frame (queries) and those in the previous frames (keys and values) conditioned on an object mask. STM-VOS boasted significant performance and runtime improvements over existing methods, and its general approach of computing pixel-to-pixel correspondences using an attention mechanism served as the basis for several subsequent works. Seong *et al.* (2020) proposed 'Kernelized Memory Networks' (KMN-VOS) which extends STM-VOS by adding spatial bias to the memory read operation by using a 2D Gaussian weighted kernel centered around the object in the past frames. Yang *et al.* (2020) proposed a more embedding-centric approach that learns per-pixel and object-level embeddings for both foreground (the target object) and background. These embeddings are then aggregated to regress the object mask in the current frame.

Although the above-mentioned approaches achieve much faster runtime compared to early VOS methods requiring online finetuning, a separate forward pass is typically needed for each target object. To remedy this, Yang *et al.* (2021b) proposed AOT, which efficiently deals with multiple objects by using learned object ID embeddings to condition mask prediction on a given object. With this approach, the runtime is largely

independent of the number of objects in the video. Cheng *et al.* (2021c) proposed STCN, which computes the pixel-wise affinities between queries and keys (current frame and past frames, respectively) in an object-agnostic manner. Subsequently, only a lightweight value encoder and mask decoder have to be run on a per-object basis to regress masks. Cheng and Schwing (2022) later extended this approach to effectively perform long-range video object segmentation. Their architecture, called XMem, is able to reason over a long temporal history by using a three-tier memory structure to encode short-term, mid-range, and long-term context from past frames.

2.4 Datasets and Benchmarks

In the preceding sections, we discussed a plethora of approaches and methods for tackling video segmentation and object-tracking tasks. It is therefore not surprising that there are also a large number of datasets and benchmarks that researchers employ to quantitatively compare their approaches to existing works. Although the evolution of datasets and benchmarks over time is closely correlated with the advancement of methods capable of tackling these tasks, we nonetheless dedicate a separate section to discuss the dataset landscape for the sake of clarity.

Salient Region Segmentation. Early datasets involving pixel-precise segmentation such as ViSal (Wang *et al.*, 2015), FBMS (Ochs *et al.*, 2013), SegTrack (Li *et al.*, 2013; Tsai *et al.*, 2012), and DAVIS 2016 (Pont-Tuset *et al.*, 2016) were motivated from the perspective of segmenting the 'salient' or 'foreground' regions in a video. Naturally, saliency and foreground were often defined as objects belonging to everyday categories that are undergoing motion across the video. These early datasets were generally small; typically containing a few dozen videos with annotations for a handful of object classes.

MOT Datasets. At the same time, several box-level Multi-object Tracking (MOT) datasets emerged, and these were generally larger in size compared to the aforementioned datasets. KITTI (Geiger *et al.*, 2012) was one of the first large-scale driving scene datasets with data and annotations not only for video-level MOT, but a diverse set of tasks such as LiDAR segmentation, depth estimation, etc. The MOTChallenge (Leal-Taixé *et al.*, 2015; Milan *et al.*, 2016) is another widely popular MOT dataset containing videos of pedestrians in various public spaces. Different from these two, the TAO (Dave *et al.*, 2020) dataset contains a diverse variety of in-the-wild videos which include driving scenes, but also indoor, outdoor, and scripted movie scenes.

VOS Datasets. With the advent of deep learning-based approaches, dataset sizes and annotation diversity for datasets with pixel-precise mask annotations rapidly increased since these approaches require a large amount of training data to prevent overfitting. Moreover, the focus shifted from low-level tasks such as motion and saliency segmenta-

tion to object-centric tasks such as VOS and VIS. The DAVIS 2017 dataset (Pont-Tuset *et al.*, 2017) improved the 2016 variant by adding more videos and also providing separate per-object masks instead of a single, combined foreground mask. Later, YouTube-VOS (Xu *et al.*, 2018b) was released which, similar to DAVIS, contains general, in-the-wild videos from the internet, but is much larger with over 3,400 videos compared to $\sim$ 150 in DAVIS. YouTube-VOS and DAVIS have since become the de-facto standard in the VOS community for evaluating new approaches.

VIS Datasets. As mentioned in Sec. 2.2.3, the task of pixel-precise multi-object tracking i.e. VIS and MOTS is relatively recent. Voigtlaender *et al.* (2019b) proposed the KITTI-MOTS and MOTSChallenge datasets, which include a subset of videos from the KITTI and MOTChallenge datasets for which the box-level annotations were converted to segmentation masks. At around the same time, more autonomous driving datasets emerged such as Waymo (Sun *et al.*, 2020) and BDD100k (Yu *et al.*, 2020) which contain object mask labels for street scene videos captured from moving vehicles in various North American cities. Simultaneously, several datasets with general, in-the-wild videos also emerged. YouTube-VIS (Yang *et al.*, 2019a) contains a subset of videos from YouTube-VOS with exhaustive mask annotations for 40 object categories. Later, the Occluded VIS (OVIS) (Qi *et al.*, 2022) dataset was proposed which contains longer videos with more objects and more challenging scenes involving object occlusions. More recently, the Unidentified Video Objects (UVO) dataset (Wang *et al.*, 2021a) was released which contains over 10,000 videos, originally from the Kinetics (Kay *et al.*, 2017) dataset for video action classification, with pixel-precise masks for a diverse set of object categories.

Video Panoptic Segmentation (VPS) Datasets. Recently, the image-level task of panoptic segmentation (Kirillov *et al.*, 2019) has been extended to video. Panoptic segmentation can be seen as an extension of instance segmentation where, in addition to segmenting object instances, non-object pixels have to be assigned a semantic class label. KITTI-STEP (Weber *et al.*, 2021) and CityscapesVPS (Kim *et al.*, 2020) are two of the first datasets for video-level panoptic segmentation, but they only contain a small number of video sequences, all of which involve driving scenes. Recently, the VIPSeg dataset (Miao *et al.*, 2022) was introduced which contains several thousand general, in-the-wild videos, and annotations for a large set of 125 semantic/object classes.

3 Preliminaries

Subsequent chapters of this thesis assume that the reader is familiar with recent, popular deep learning techniques as well as certain concepts related to video segmentation and multi-object tracking. This chapter recaps multiple concepts which will be repeatedly used in later chapters and hence, it may be useful for readers to go over them.

We will first explain the architecture behind residual networks, better known as 'ResNet', since these are commonly used as backbone networks for extracting multi-scale features from images. Additionally, we will recap 3D extensions of ResNet which are used for various video understanding tasks. Next, we will discuss transformer networks and multi-head attention since these have had a huge impact on computer vision in recent years and are extensively used in Chapters 5 and 7. Toward the end, we will discuss evaluation measures that are commonly used for video segmentation with a focus on their merits, demerits, and the motivation behind their formulation.

3.1 ResNet

Residual **Net**works were first proposed by He *et al.* (2016). Prior to this, Convolutional Neural Network (CNN) architectures such as VGGNet (Simonyan and Zisserman, 2015) were limited to a few dozen layers because it was challenging to effectively train deeper networks. These architectures were based on the earlier AlexNet architecture (Krizhevsky *et al.*, 2012) and comprise several 'layers' stacked on top of each other. Each layer, in turn, involves a convolution (usually with a 3×3 kernel) followed by a non-linear activation function e.g. ReLU (Rectified Linear Unit), and optionally, a pooling layer.

He *et al.* (2016) observed that the performance of such architectures could be boosted by adding more layers, but only up to ≈ 30. Beyond this point, the performance

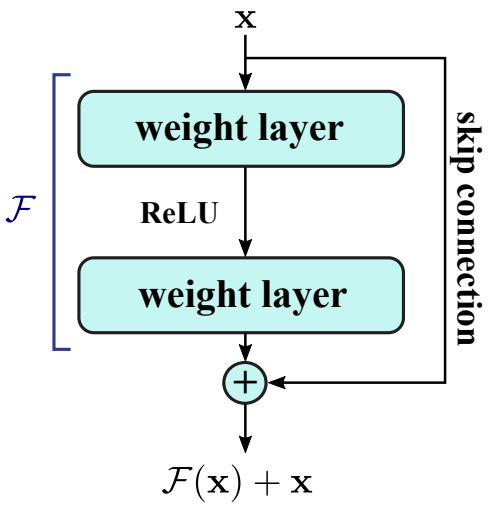


Figure 3.1: Illustration of a Residual Block. This is a slightly modified version of Fig. 2 in He *et al.* (2016).

saturates and then starts to reduce. What is worth noting is that this is not due to overfitting, because both test-time and train-time errors follow the same pattern. Rather, it was shown that this behavior stems from regular occurrences of vanishing or exploding gradients during the training process. To overcome this, He *et al.* (2016) proposed adding residual paths to the forward pass where the input to the i-th layer of the network comprises not only the output from the preceding layer $i-1$, but also the output from layer $i-N$ (where $N>1$) via a 'skip connection'.

The motivation for doing so becomes clear from a mathematical perspective: a typical CNN layer (without skip connections) is optimized to learn a mapping $\mathcal{H}(\mathbf{x})$ to transform the feature activation $\mathbf{x}$ from the previous layer such that the final training loss is minimized. On the other hand, the output for a layer with a skip connection can be written as $\mathcal{F}(\mathbf{x}) + \mathbf{x}$ where $\mathcal{F}$ is a residual function that only has to predict a suitable offset that is applied to the input $\mathbf{x}$. The idea is that, compared to $\mathcal{H}$, learning $\mathcal{F}$ is a more tractable optimization objective for the gradient descent-based approaches used to train CNNs.

Building on this, He *et al.* (2016) proposed a CNN architecture which is now known as 'ResNet'. The fundamental building block for this is the 'Residual block' which is illustrated in Fig. 3.1. It contains two convolutional layers with an intermediate ReLU activation and a skip connection between the input and output. The overall network comprises several such residual blocks stacked on top of one another with intermittent pooling operations to reduce the size of the image feature map. He *et al.* (2016) experimented and created multiple variants of this architecture with different numbers of layers and showed that performance gains were achievable up to 152 layers which was a major improvement over the pre-existing VGGNet which contained at most 19 layers.

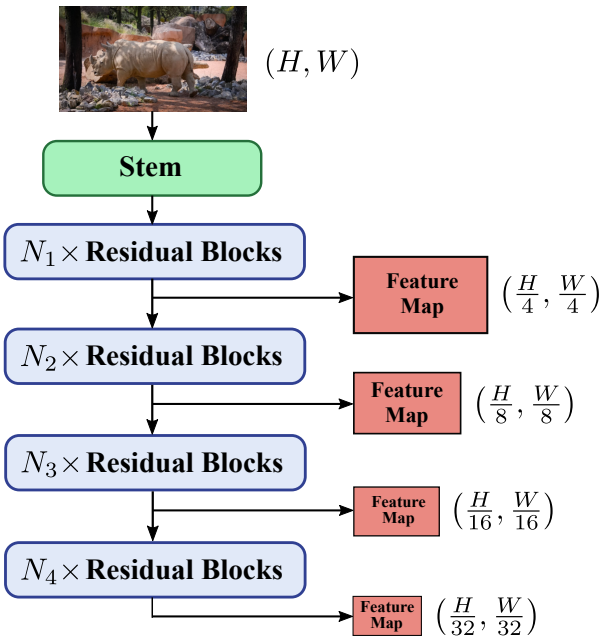


Figure 3.2: Illustration of a ResNet model used as a backbone image feature extractor.

3.1.1 Feature Extraction Backbone

In the original publication, ResNet was used for the task of image classification on the ImageNet (Deng *et al.*, 2009) benchmark where it outperformed existing approaches at the time. However, the scope of this architecture extends well beyond this task because a number of downstream tasks, including those involving dense prediction and segmentation, employ a pre-trained ResNet as a backbone network for extracting meaningful features from images. In fact, we employ some variant of ResNet for all the network architectures proposed in this thesis (Chapters 4, 5, and 7).

For dense prediction tasks such as ours, the goal of such backbone networks is to obtain useful features for the input image at multiple spatial scales. The idea here is that features at smaller scales encode high-level, holistic information about the image, whereas features at larger scales carry fine-grained information about object location and boundaries. Hence, we build architectures for downstream segmentation tasks such that they can leverage useful information from multiple scales.

Fig. 3.2 coarsely illustrates the architecture of a ResNet and how it is typically used for extracting multi-scale features for a given image with dimensions (H, W). Here we conceptually divide the architecture into five stages[1]. Stage 0 is the 'stem' which contains a 7×7 convolution followed by a max-pooling layer, both with stride 2, i.e., the feature map output by the stem is $4\times$ downsampled w.r.t the input image. The four

[1]The word 'stage' and the numbering convention for it used in Fig. 3.2 is consistent with the de-facto community standard that is seen in popular deep learning libraries and open-source projects.

subsequent stages contain $N_1, \ldots, N_4$ residual blocks, respectively. Note that the very first convolutional layer in stages 2, 3, and 4 has a stride of 2 which downsamples the input. Multi-scale feature maps are obtained by simply taking the output from each of the stages, thus yielding 4 feature maps at the 4×, 8×, 16×, and 32× downsampled scales. Note that at the end of the network, we omit the average pooling and subsequent fully connected layer since those are only needed for classification.

Variant	N_1	N_2	N_3	N_4	Layers per Residual Block
ResNet-18	2	2	2	2	2
ResNet-34	3	4	6	3	2
ResNet-50	3	4	6	3	3
ResNet-101	3	4	23	3	3
ResNet-152	3	8	36	3	3

Table 3.1: Depths of the various stages of ResNet for different variants of the architecture.

He *et al.* (2016) proposed multiple variants of ResNet with different depths. Variants with more layers perform better, but also incur a larger memory footprint, have slower runtimes, and are more susceptible to overfitting. Table 3.1 summarizes the per-stage depths for the five most popular ResNet variants. Note that the Residual Block in Fig. 3.1 contains two layers, but for the 50, 101, and 152 layer variants, a modified Residual Block with 3 layers is used. For segmentation tasks, ResNet-50 and ResNet-101 are popular choices for backbone networks since they offer a good compromise between performance and computational efficiency.

3.1.2 3D ResNet

CNNs such as the ResNet architecture discussed above involve 2D convolution operations that are applied to a 2D feature map with some height and width (note that here we do not consider the feature size as a dimension). Although CNNs with 2D convolutions receive most of the research focus, there is a research sub-community that applies CNNs with 3D convolutions for video tasks. This idea is that since 2D convolutional networks are successful for image-level tasks, it is logical to think that 3D convolutional networks can do the same for video-level tasks. Another advantage of processing video in this way is that it enables network architectures to be more compact since no explicit mechanism is needed for temporal reasoning. Rather, this happens implicitly when the 3D convolutional kernels are optimized for the final loss function. Before discussing further, it is worth clarifying that here we are using 3D convolutions in the context of video understanding, and not for tasks related to 3D point clouds, which is by far a

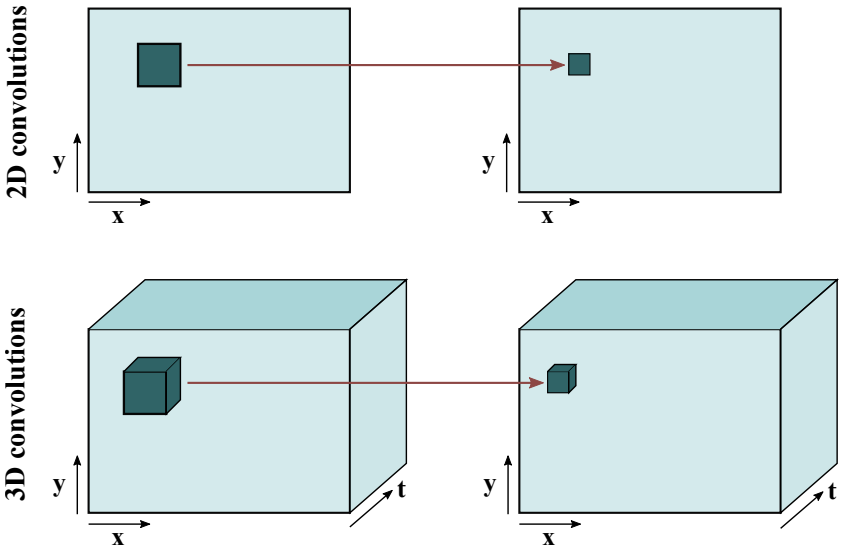


Figure 3.3: Visual comparison of 2D and 3D convolutions in the context of image and video understanding, respectively. Whereas the input and convolutional kernels in 2D convolutions span the spatial (x, y) dimensions, the input and kernels in 3D convolutions span the spatial (x, y) and temporal (t) dimensions. Note that the feature dimension is ignored in this diagram.

more common use case for 3D convolutions. Fig. 3.3 illustrates the difference between the two types of convolution.

Early works (Hara *et al.*, 2017; Ji *et al.*, 2012; Tran *et al.*, 2015; Varol *et al.*, 2017) for video action recognition regularly employed 3D CNNs. The architectures for these networks closely resemble their 2D CNN counterparts at the time with the major difference that the convolutional kernels usually have size $3 \times 3 \times 3$ (i.e. 3-dimensional). One drawback, however, is that the computational and memory footprint of these models is significantly higher than that of their 2D counterparts. To remedy this, Tran *et al.* (2018) proposed so-called 'R(2+1)D' convolutions which factorize the vanilla 3D convolutional into two operations that go over the spatial and temporal dimensions, respectively. Later, Tran *et al.* (2019) proposed using channel-separated 3D convolutions (also called dilated convolutions) which have a significantly smaller memory footprint without compromising performance.

Although 3D convolutions have been extensively utilized for video classification, their use for dense prediction tasks is, by comparison, much more limited. In Chapter 4, we will show that 3D convolutions can also be effectively applied for object segmentation in video.

3.2 Multi-head Attention

As discussed earlier, the emergence of AlexNet (Krizhevsky *et al.*, 2012) ushered a paradigm shift in computer vision, and, during the course of the following years, researchers developed CNN-based architectures for tackling a plethora of tasks. However, almost all of these approaches utilized the same fundamental tools for feature learning: convolutional layers and MLPs comprising fully connected layers. In 2020, however, this changed with the emergence of the 'Vision Transformer' (ViT) (Dosovitskiy *et al.*, 2021) and the DETR (Carion *et al.*, 2020) architecture for image-level object detection. These architectures were unique because they employed 'Multihead-Attention' (MHA) which was already popular in the Natural Language Processing (NLP) domain but was seldom used in computer vision methods up until this time.

3.2.1 Formulation

Multi-head attention was first proposed by Vaswani *et al.* (2017). Although it was applied for the task of language translation, the operation is itself generic and can be applied in any setting that involves feature learning. We will first discuss the mathematical formulation and then discuss how it is applied to various use cases.

Given three sets of features, namely queries $\mathbf{Q} \in \mathbb{R}^{M \times D}$, keys $\mathbf{K} \in \mathbb{R}^{N \times D}$ and values $\mathbf{V} \in \mathbb{R}^{N \times D}$, the attention operation produces an output $\mathbf{Y} \in \mathbb{R}^{M \times D}$ (which has the same dimensions as $\mathbf{Q}$) as follows:

$$\mathbf{Y} \leftarrow \texttt{softmax}\left(\frac{W_{\text{q}}\mathbf{Q} \cdot W_{\text{k}}\mathbf{K}^T}{\sqrt{D}}\right) \cdot W_{\text{v}}\mathbf{V} \tag{3.1}$$

Here, W_{q}, W_{k} and W_{v} are learned weights $\in \mathbb{R}^{D \times D}$ for linear projections that are applied to $\mathbf{Q}$, $\mathbf{K}$ and $\mathbf{V}$, respectively. The term $W_{\text{q}}\mathbf{Q} \cdot W_{\text{k}}\mathbf{K}^T \in \mathbb{R}^{M \times N}$ is sometimes called the 'attention weight', and is intuitively an affinity matrix that encodes the correlation between the query and key features. The subsequent softmax is applied over the N-sized dimension i.e. the columns. $\sqrt{D}$ is a scalar representing the softmax temperature. The resulting normalized attention weights are then multiplied with the values $W_v\mathbf{V}$ to obtain the output $\mathbf{Y}$ i.e. the output is a weighted sum of the values. Note that Eq. 3.1 describes basic, single-head attention. For multi-head attention, we split the operation into h parallel operations along the D-sized dimension. Hence, the queries, keys, and values would then have sizes $\mathbf{Q} \in \mathbb{R}^{M \times \frac{D}{h}}$, $\mathbf{K} \in \mathbb{R}^{N \times \frac{D}{h}}$ and $\mathbf{V} \in \mathbb{R}^{N \times \frac{D}{h}}$, respectively, the weight matrices would have size $\mathbb{R}^{\frac{D}{h} \times \frac{D}{h}}$, and finally, the softmax temperature would also change to $\sqrt{D/h}$. At the end, the h per-head outputs are concatenated to obtain the final D-dimensional output. The idea behind this is to have h different projections of the

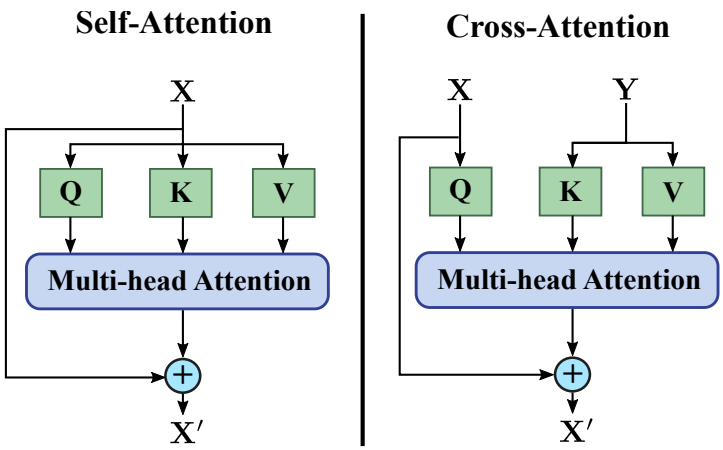


Figure 3.4: Comparison of self-attention and cross-attention. In practice, several normalization and linear layers are applied before and after the multi-head attention operation, but here these are omitted for simplicity.

input features which gives the network more flexibility in modeling feature interactions between the queries and keys/values.

At this point, it may still be unclear to the reader what the queries, keys, and values actually represent when multi-head attention is practically employed in a network architecture. This mainly depends on the use case. Two of the most common use cases that are widely applied in literature are explained below.

3.2.1.1 Self-Attention

Self-attention refers to the scenario where the queries, keys, and values all originate from the same set of features. Let us consider an example where $X \in \mathbb{R}^{N \times D}$ denotes the flattened D-dimensional feature map for an input image. In this setting, we can apply multiple self-attention operations to X in order to enhance its feature representation, similar to how a ResNet applies back-to-back convolution operations. Hence, in this setting, multi-head attention can be seen as a computational workhorse that substitutes convolution. It is, however, worth noting that the attention operation has key mathematical differences to convolutions.

When applying self-attention to X, the queries, keys, and values are all derived from X and hence, have the same size i.e. $M = N$ in Eq. 3.1. In the attention operation, the attention weight $W_{\mathrm{q}}\mathbf{Q} \cdot W_{\mathrm{k}}\mathbf{K}^T$ is an $N \times N$ matrix that encodes the affinity between every possible pair of features. The subsequent softmax and multiplication with V yields an output which is a weighted sum of all the features based on their respective affinities. In general, the output X' of an 'attention layer' involves adding the actual output of the attention operation as a residual to the input features X:

$$X' \leftarrow X + \texttt{SelfAttention}(X) \tag{3.2}$$

3.2.1.2 Cross-Attention

The second use-case for multi-head attention is called cross-attention and is also more relevant for this thesis. Here we have a set of features $X \in \mathbb{R}^{M \times D}$ which we want to update by "attending to" another set of features $Y \in \mathbb{R}^{N \times D}$. In other words, this can be seen as a unidirectional operation where X absorbs information from Y. The updated features X' are therefore obtained as follows:

$$X' \leftarrow X + \texttt{CrossAttention}(X, Y) \tag{3.3}$$

In this case, the queries originate from X whereas the keys and values originate from Y. Cross-attention serves as an easy-to-implement but powerful tool for enabling information flow between two heterogeneous sets of features. This is not possible with convolution operations which expect the input to have a grid-like structure. In the domain of image/video segmentation, cross-attention is heavily used to enable network architectures to represent objects using concise feature vectors, sometimes called 'queries'. This is achieved by applying multiple cross-attention operations in an iterative fashion wherein the queries attend to the image/video features and thus learn to encode the objects present in the image/video.

Fig. 3.4 briefly illustrates both self- and cross-attention operations to highlight the difference between them.

3.2.2 Positional Encodings

Image and video data have an inherent spatial (and temporal) structure, and therefore, any learning task based on this data should leverage this structure. However, the attention operation given in Eq. 3.1 is permutation-invariant with respect to the keys and values. To mitigate this shortcoming, positional encodings are commonly used to introduce inductive bias to the attention operation by adding spatial coordinate information. These were initially proposed by Vaswani *et al.* (2017) for NLP to impart temporal indices to word embeddings. Subsequently, works by Parmar *et al.* (2018), Bello *et al.* (2019), and Dosovitskiy *et al.* (2021) proposed variants of positional encodings for image data. The encodings can be applied to the keys and/or the queries depending on which of these are derived from image data. We denote the encodings for keys with $E_k \in \mathbb{R}^{N \times D}$ and those for queries with $E_q \in \mathbb{R}^{M \times D}$. The encodings have the same size as the keys/queries and are typically applied by addition. Thus, Eq. 3.1 would be modified as follows:

$$\mathbf{Y} \leftarrow \texttt{softmax}\left(\frac{W_{\text{q}}(\mathbf{Q}+\mathbf{E_q})\cdot W_{\text{k}}(\mathbf{K}+\mathbf{E_k})^T}{\sqrt{D}}\right)\cdot W_{\text{v}}\mathbf{V} \tag{3.4}$$

Note that with this formulation, positional information is used to calculate the attention weights which in turn affect with weights used when summing over the value, but the positional coordinates do not not directly manifest themselves in the output $\mathbf{Y}$.

3.2.2.1 Fixed Sinusoidal Encodings

These are non-parametric encodings where the encoding $e \in E \in \mathbb{R}^D$ for the image feature at coordinates (x, y) is generated by applying sine and cosine activations with varying frequencies to each of the coordinates. For D-dimensional encodings we typically dedicate $\frac{D}{2}$ dimensions for encoding the x and y dimensions, respectively. For each coordinate, $\frac{D}{4}$ dimensions employ the sine activation at varying frequencies whereas the remaining $\frac{D}{4}$ apply cosine activations. Thus, overall the positional encodings are a combination of sine and cosine functions applied to both coordinates at multiple frequencies.

3.2.2.2 Learned Encodings

These are parametric encodings that are used by architectures such as ViT (Dosovitskiy *et al.*, 2021) where the inputs to the attention operation always have a fixed and known size (i.e. N and M are network design parameters). For this setting, we can initialize a set of learned encodings randomly and optimize them during training. The advantage here is that the encodings are learned in an end-to-end fashion, but the drawback is that such a formulation is not feasible when working with variable-sized image inputs, as is usually the case for segmentation tasks. Thus, the works presented in Chapters 5 and 7 exclusively employ fixed sinusoidal encodings.

3.3 Transformer Networks

Having covered the basics related to multi-head attention, we are now in a position to discuss Transformer networks. In the context of deep learning, Transformers are a family of network architectures that employ successive layers of multi-head attention and MLPs to learn features for optimizing the given training objective. In the seminal work by Vaswani *et al.* (2017), the transformer network was composed of an encoder and decoder, as illustrated in Fig. 3.5. This network was applied to the task of language translation, and so the 'Input Embeddings' are word embeddings corresponding to the input text. The encoder refines the feature representation for these embeddings in an iterative fashion by passing them through L layers; each layer comprises a self-attention

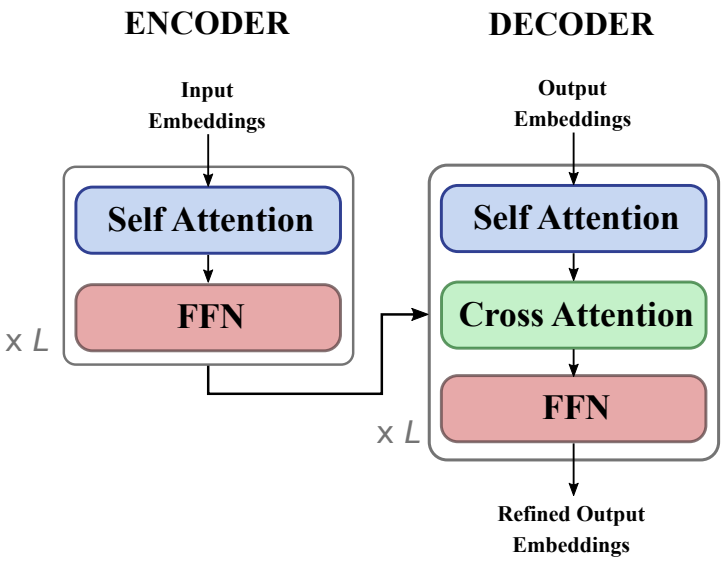


Figure 3.5: High-level architecture of a basic transformer network. We omit the classification step, normalization layers, and positional encodings for simplicity. **FFN:** Feed-forward Network, **L:** number of encoder/decoder layers.

operation where the embeddings exchange information among each other, followed by a Feedforward Network (FFN) which is applied individually to each embedding.

Similar to the input, each word in the output text is also represented by a word embedding. The input to the decoder is therefore a set of embeddings representing the translated text. The feature representation of these embeddings is refined by applying them to several layers, each of which contains: (1) a self-attention operation where the output embeddings exchange information among themselves, (2) a cross-attention operation where the queries are derived from the output embeddings, and they keys/values are derived from the input text embeddings produced by the encoder, and (3) a Feedforward Network that is applied individually to each output embedding. The idea is that the output embeddings interact with each other via self-attention, and also learn about the input text by cross-attending to the embeddings for the input text produced by the encoder. Obtaining the final translated text involves a classification step, but we omit the discussion for this here since it is not directly relevant to our usage of transformers in this thesis.

Although transformer networks have nowadays become ubiquitous in computer vision and are used for several different applications, the underlying architectural building blocks still closely resemble those shown in Fig. 3.5. In the following subsections, we will discuss four popular network architectures that are highly relevant for upcoming chapters. Three of these (DETR, Deformable-DETR, and Mask2Former) perform object detection and segmentation whereas the fourth (Swin) is a generic backbone for learning image features.

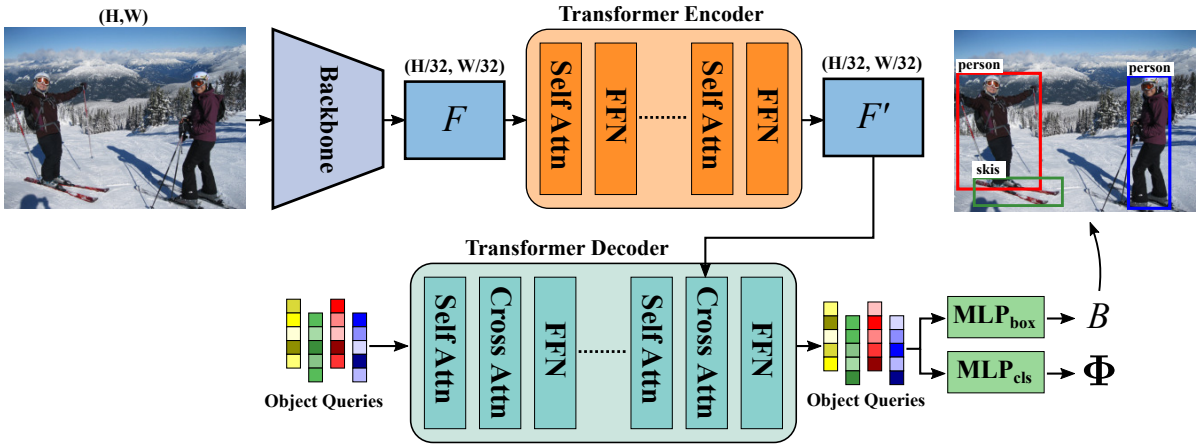


Figure 3.6: Illustration of the DETR architecture.

3.3.1 DETR

Carion *et al.* (2020) proposed DETR, which is a transformer-based end-to-end approach for object detection in images (this task involves predicting bounding-box coordinates for all objects in an image belonging to a set of pre-defined categories). Until this time, state-of-the-art approaches for object detection were largely based on Mask R-CNN (He *et al.*, 2017). Although Mask R-CNN is highly effective, it is a 'two-stage' detector where the first stage predicts a large set of class-agnostic object proposals, followed by a second stage which receives image feature crops of these proposals as input and performs classification and mask regression. This workflow results in slow run times and also involves several hyper-parameters for generating object proposals and performing Non-Maximum Suppression (NMS) on the output predictions.

DETR mitigates these issues by adopting a single-stage end-to-end approach that leverages the transformer network's flexibility of enabling interactions between heterogeneous feature sets. Specifically, DETR represents each object prediction with a D-dimensional 'query' feature. Overall, a fixed number (M) of such queries are initialized randomly at the start of training. Typically, $M = 100$ and is chosen such that it is an upper bound on the number of objects in an image. We will denote these queries with $Q \in \mathbb{R}^{M \times D}$. In the following paragraphs, we will discuss the components of the DETR architecture (illustrated in Fig. 3.6) as well as the training and inference procedure.

3.3.1.1 Backbone

The input image of size (H, W) is input to an ImageNet pretrained backbone network to learn useful features. For this, a ResNet (He *et al.*, 2016) is employed which produces a $32\times$ downsampled D-dimensional feature map which we denote with $F \in \mathbb{R}^{\frac{H}{32} \times \frac{W}{32} \times D}$.

3.3.1.2 Transformer Encoder

This component has a similar architecture to the encoder illustrated in Fig. 3.5 where the idea is to improve the feature representation in F by iteratively applying self-attention. In some ways, the encoder can also be seen as a conceptual extension of the backbone since it learns image-level features, however, it is worth noting that self-attention enables global feature interaction, i.e. every pixel feature interacts with every other pixel feature. This is in contrast to the convolution or sliding window attention operations in the backbone which only allow feature interactions within a certain window. We denote the output of the transformer encoder with $F' \in \mathbb{R}^{\frac{H}{32}, \frac{W}{32}}$.

3.3.1.3 Transformer Decoder

The transformer decoder is the main component that tackles the object detection task. The inputs to the decoder are the object queries $Q \in \mathbb{R}^{M \times D}$, and the feature map F' output by the encoder. Similar to the seminal transformer network architecture, the decoder in DETR comprises several layers, each of which contains self-attention, cross-attention, and a feedforward network. The decoder updates the queries Q (which are statically initialized) by iteratively applying self-attention which allows queries to interact with each other, followed by cross-attention where the queries learn scene context by attending to the image features F'. The output of the decoder is a set of updated object queries which we denote with $Q' \in \mathbb{R}^{M \times D}$. The underlying idea is that this iterative attention enables the queries to learn correspondences to the objects in the input image.

3.3.1.4 Box Regression and Classification

Finally, Q' is applied to two MLPs. MLP_{box} regresses bounding box predictions $B \in \mathbb{R}^{M \times 4}$ where the boxes are parameterized by a 4D vector which specifies the height, width and (x, y) coordinates of the center of the bounding box. On the other hand, MLP_{cls} predicts classification scores $\Phi \in \mathbb{R}^{M \times (C+1)}$ where C is the number of object categories specified by the dataset. Φ contains one extra prediction for a so-called 'no-object' class. This is required because the number of queries (M) is fixed and is assumed to be an upper bound on the number of objects that we expect to encounter in a given image. The number of actual objects is therefore usually smaller than M, and the no-object class allows the network to 'switch off' queries that do not correspond to any object by assigning a high classification score for the no-object class.

3.3.1.5 Training

One challenge with training DETR is that there is no user-enforced assignment between the query predictions and the set of ground-truth objects in the image. To supervise the network during training, Carion *et al.* (2020) associate predictions with ground truth by applying Hungarian Matching wherein the association score is a weighted sum of two components: the IoU between the predicted and ground-truth boxes, and (2) the predicted classification score for the ground-truth class ID. Queries that are not associated with any ground-truth object are supervised to have a no-object class label and their corresponding box coordinates are ignored.

3.3.1.6 Differences to Existing Approaches

DETR proved to be highly popular and formed the basis for several subsequent works in both image and video segmentation. The main advantage of this approach is that it leverages an abstract representation for the objects (i.e. the queries) which implicitly learn correspondences to objects in the input image solely through the final loss supervision. This is in stark contrast to existing approaches at the time which were largely based on Mask R-CNN (He *et al.*, 2017) and YOLO (Redmon *et al.*, 2016). Both families of methods used a very large, hand-crafted set of 'anchor boxes' which serve as coarse object proposals that are later regressed to fit objects in the scene. This results in duplicated predictions for the same object which are removed by applying NMS which is a slow and costly post-processing step. DETR, on the other hand, requires no such post-processing since the queries implicitly learn non-overlapping correspondences to the image objects.

There were, however, some drawbacks of this approach. Firstly, DETR requires significantly longer to converge. On the COCO dataset (Lin *et al.*, 2014), Mask R-CNN is typically trained for 18-50 epochs, but DETR requires 300 epochs to fully converge. Moreover, the transformer encoder can only be applied to the $32\times$ downsampled feature map since the memory requirement for self-attention scales quadratically with image size and would therefore be infeasible for larger feature maps. As a consequence, DETR performs worse than existing methods for detecting small-sized objects.

3.3.2 Deformable DETR

Zhu *et al.* (2020) proposed as an extension to DETR which mitigates both of the above-mentioned limitations. The main contribution of their work is the 'Deformable Attention' operation (illustrated in Fig. 3.7) whereas the remaining network architecture is largely similar to DETR.

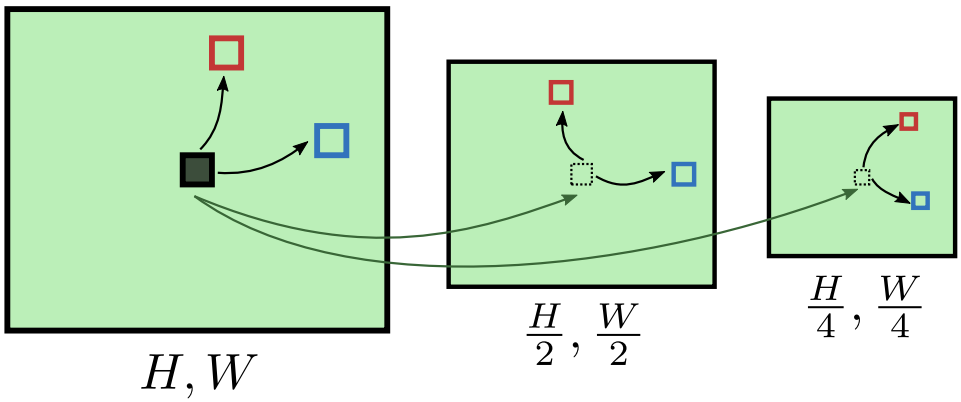


Figure 3.7: Illustration of deformable self-attention being applied to feature maps at 3 different scales. The attention operation is shown for when the feature in the black square in the largest feature map is the query. The keys and values are obtained by sampling the features at two learned offsets for each of the three scales.

Deformable Attention mitigates the large memory footprint incurred by vanilla attention, particularly for the self-attention operations in the transformer encoder. Let $\mathbf{X} \in \mathbb{R}^{N \times D}$ be the D-dimensional feature map for an image. The attention weight matrix thus has size $N \times N$ since both the queries and the keys contain N features. This makes it infeasible to apply self-attention to large-sized feature maps. Deformable attention mitigates this problem by instead using a smaller subset of N' features from $\mathbf{X}$ for the keys and values where $N' \ll N$. This subset of features is chosen based on learned positional offsets: for a given query feature at coordinates (x, y), an MLP is used to predict a set of N' normalized offsets which are applied to (x, y) to obtain the coordinates of the features which will be used as keys and values. Bilinear interpolation is used to sample these features since the resulting coordinates are generally not integers. If the sampled coordinates lie outside the feature map boundaries then all-zeros features are returned.

Note from Fig. 3.7 that the keys and values are also sampled from multiple feature maps at different scales. Thus, given S scales, the MLP actually predicts $S \times N'$ offsets which are applied to the normalized query point coordinates across all S scales. For the example in the figure, $S = 3$ and $N' = 2$.

Lastly, we point out that Deformable DETR also employed deformable cross-attention (not illustrated in Fig. 3.7). This was done by applying the object queries to an MLP to predict the $S \times N'$ offsets from which the keys and values were sampled.

3.3.2.1 Advantages over DETR

Due to the above-mentioned formulation, the memory footprint for Deformable Attention is considerably lower than that for vanilla attention. Note that even though

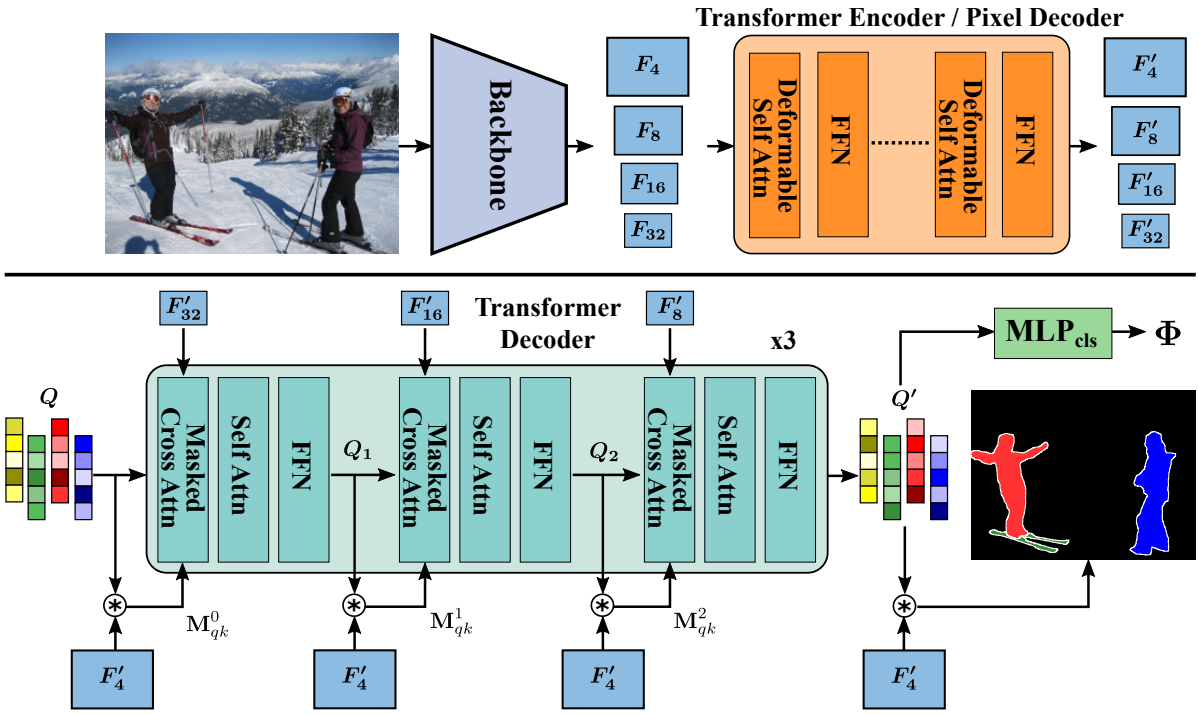


Figure 3.8: Illustration of Mask2Former (Cheng *et al.*, 2022) architecture.

deformable attention attends to fewer keys and values, the operation still preserves globality since the sampled features can lie anywhere on the feature map. Moreover, the formulation allows feature interaction across multiple scales which is helpful in pooling information about high-level semantics (which are better represented in smaller feature maps) with fine-grained object boundary information (which is better preserved in larger feature maps). In this sense, the transformer encoder in Deformable DETR can be seen as a substitute for Feature Pyramid Networks (FPN) (Lin *et al.*, 2017) which have been extensively used as an add-on block to backbone networks to pool features over multiple spatial scales.

Overall, Deformable DETR achieves better performance than DETR, particularly for detecting smaller objects since it is able to leverage feature maps at the $8\times$ downsampled scale compared to the $32\times$ scale used by DETR. Furthermore, Deformable DETR also converges significantly faster: on the COCO dataset, it outperforms DETR after training for 50 epochs, compared to the 300 epochs required by the latter. Zhu *et al.* (2020) argue that the lower number of key and value features simplify the learning objective and hence, accelerate convergence.

3.3.3 Mask2Former

Mask2Former (Cheng *et al.*, 2022) can be seen as a continuation of DETR and Deformable DETR since it draws inspiration from both. One key difference, however, is that Mask2Former produces pixel-precise segmentation masks whereas the two aforementioned methods only perform object detection i.e. bounding-box level prediction. Moreover, Mask2Former was proposed as a generalized method that can perform any image segmentation task including instance segmentation, semantic segmentation, and panoptic segmentation.

The network architecture is illustrated in Fig. 3.8. The overall structure is very similar to DETR (Fig. 3.6) and Deformable DETR. The input image is applied to a backbone network that produces multi-scale feature maps $\{F_i \,|\, i = 4, 8, 16, 32\}$ where i denotes the downsampling factor with respect to the input image size. These feature maps are input to a 'Pixel Decoder' which is almost identical to the Transformer Encoder in Deformable DETR: a series of deformable self-attention operations are applied to the feature maps across multiple scales. The output of the Pixel Decoder $\{F'_i \,|\, i = 4, 8, 16, 32\}$, together with a set of statically initialized queries $Q \in \mathbb{R}^{M \times D}$, are then input to a Transformer Decoder where the queries iteratively attend to each other via self-attention, and to the image features via cross-attention. The updated queries $Q' \in \mathbb{R}^{M \times D}$ are then applied to MLP_{cls} which produces classification scores $\Phi \in \mathbb{R}^{M \times C+1}$ for each of the C object/semantic categories in the dataset, and an additional 'no-object' class for inactive queries.

In the subsections below, we highlight the key changes and contributions of the Mask2Former architecture.

3.3.3.1 Mask Prediction

To predict segmentation masks, Cheng *et al.* (2022) compute the dot-product between the F'_4 (the highest resolution feature map) and the queries Q' output by the transformer decoder. This yields a set of mask logits with shape $(M, \frac{H}{4}, \frac{W}{4})$ which are upsampled to the full image resolution and then thresholded at 0 (equivalent to thresholding at 0.5 after applying a sigmoid activation) to obtain the final binary masks. During training, the mask predictions are supervised by a combination of binary cross-entropy and soft-IoU (DICE) losses.

3.3.3.2 Masked Cross-Attention at Multiple Scales

Whereas DETR and Deformable DETR employed vanilla attention and deformable attention for cross-attention, respectively, Mask2Former proposed a new approach called Masked Cross-Attention wherein the attention operation is modified as follows:

$$\mathbf{Y} \leftarrow \texttt{softmax}\left(\frac{W_{\text{q}}(\mathbf{Q}+\mathbf{E_q}) \cdot W_{\text{k}}(\mathbf{K}+\mathbf{E_k})^T + \mathbf{M_{qk}}}{\sqrt{D}}\right) \cdot W_{\text{v}}\mathbf{V} \tag{3.5}$$

Here, the term in blue highlights the difference to the standard attention operation in Eq. 3.1. Specifically, $\mathbf{M_{qk}} \in \mathbb{R}^{M \times N}$ is a tensor containing the binary segmentation mask for every query at every pixel (key) location, and is obtained by taking the dot-product between F_4' and the queries output by the previous transformer decoder layer. Conceptually, this entails applying an offset $\mathbf{M_{qk}}$ to the key-query product (i.e. the attention weights). This has the effect of amplifying the key-query correspondences for pixel locations where the predicted mask from the previous layer produced a strong response. The idea is that by doing so, each query can better focus on the relevant part of the image, thus improving both convergence and performance. Furthermore, the decoder layers receive image feature maps at different scales in a round-robin fashion: $F_{32}, F_{16}, F_8, F_{32}$....

Cheng *et al.* (2022) also noted that with this architecture, it is beneficial to apply cross-attention before self-attention in the Transformer Decoder. In Fig. 3.8, $\mathbf{M_{qk}}^l$ denotes the mask used for cross-attention at the l-th decoder layer.

3.3.3.3 Universal Image Segmentation

Mask2Former was applied not just for instance (object) segmentation, but also to panoptic segmentation to highlight that it can generalize to any image segmentation task. To recap, panoptic segmentation involves segmenting and classifying objects belonging to instantiable 'thing' classes (e.g. *car*, *person*) and additionally, segmenting non-instantiable 'stuff' classes (e.g. *vegetation*, *road*). To tackle this task, Mask2Former models stuff classes with the same set of object queries and merely alters the post-processing workflow to combine all stuff class segments into a single instance. Our work in Chapter 7 seeks inspiration from this aspect of Mask2Former to develop a multi-task video segmentation approach.

3.3.3.4 Implementation Details

In addition to the above-mentioned changes, Mask2Former leveraged a number of thoughtful implementation details to improve efficiency and performance. Recall from Sec. 3.3.1 that DETR employs IoU-based Hungarian matching to compute correspondences between object queries and ground-truth instances. However, Mask2Former does pixel-precise segmentation, and therefore computing exhaustive pair-wise IoUs between all ground-truth instances and predictions is computationally infeasible. To mitigate this issue, Cheng *et al.* (2022) take inspiration from Kirillov *et al.* (2020) and compute the IoU for only a subset of points which are sampled using a combination of random and

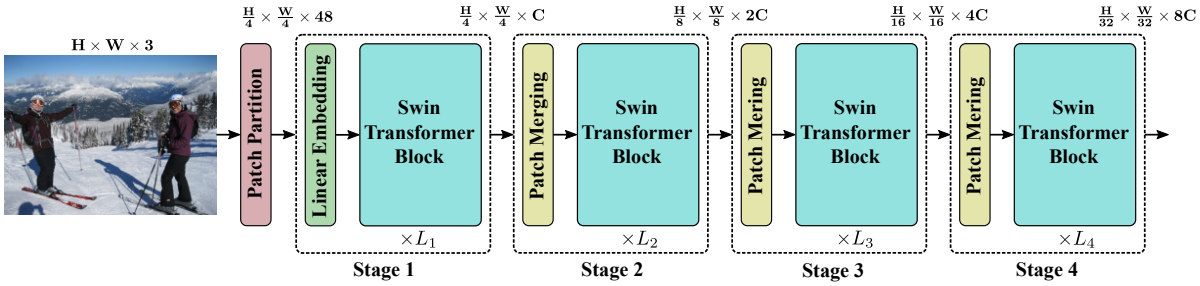


Figure 3.9: Swin Transformer (Liu *et al.*, 2021) architecture.

uncertainty-based sampling. In fact, the mask loss is also applied to a similar subset of points which greatly reduces the memory footprint.

3.3.4 Swin Transformer

The final network architecture that we will discuss here is the Swin Transformer (Liu *et al.*, 2021). Whereas the previous three methods tackle object detection/segmentation in images, the Swin Transformer is an attention-based backbone architecture that is applied to image classification. More importantly, however, it can be used as an effective image feature extractor for downstream tasks such as pixel-precise segmentation. Although ResNet is generally still popular as a backbone network, most recent state-of-the-art approaches for video segmentation tasks employ Swin Transformer and only report results with ResNet for the sake of fair comparison to older approaches.

The network architecture is illustrated in Fig. 3.9. At a high level, we note that similar to ResNet, Swin Transformer also comprises multiple stages where all but the first stage outputs a feature map with half the spatial size and double the feature channel size as its input. However, the architecture of the stages themselves is fundamentally different since it involves multi-head attention as opposed to convolutions.

The forward pass of a Swin transformer proceeds as follows: the input RGB image of size $(H \times W \times 3)$ is divided into patches, each of size 4×4, followed by a channel-wise concatenation of the pixels within a patch; this yields a tensor of size $\left(\frac{H}{4} \times \frac{W}{4} \times 48\right)$ (since $4 \times 4 \times 3 = 48$). This tensor is then input to the first of four stages. In the first stage, a linear layer is applied to increase the number of feature channels from 48 to C. The resulting feature map of size $\left(\frac{H}{4} \times \frac{W}{4} \times C\right)$ is applied to a series of L_1 'Swin Transformer Blocks', each of which contains self-attention layers, an MLP, and two LayerNorm (Ba *et al.*, 2016) operations. Subsequent stages (2-4) include a 'Patch Merging' operation at the start which reduces the spatial dimensions of the feature map by a factor of 2. This is done by concatenating the features in each 2×2 spatial neighborhood along the feature dimension, followed by a linear layer to reduce the number of feature channels

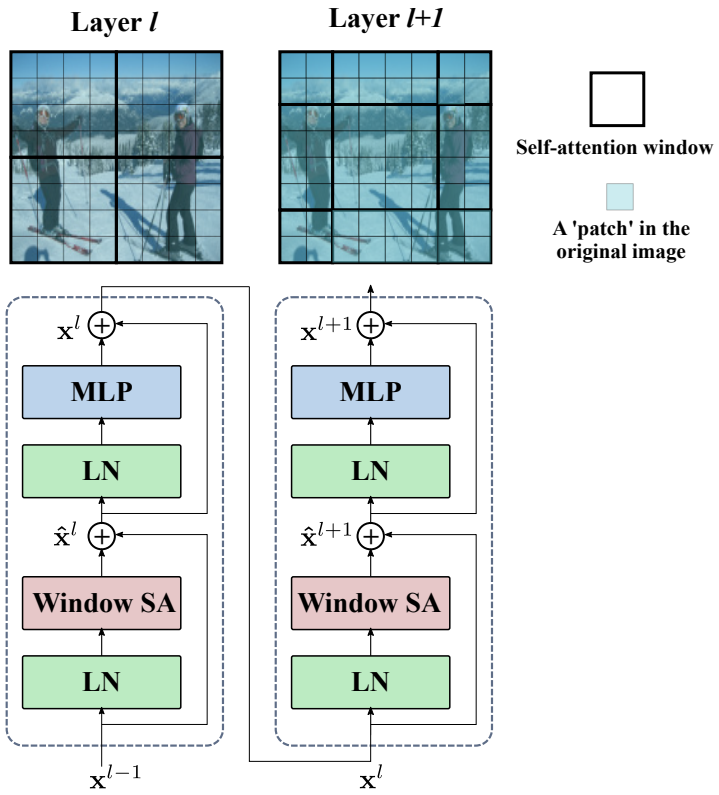


Figure 3.10: Illustration of two successive Swin Transformer Blocks along with visualization of the self self-attention windows for each block. The input to the first block is the feature map $\mathbf{x}^{l-1}$. LN: LayerNorm (Ba *et al.*, 2016).

by half. Overall, this results in a feature map with half the spatial size and double the feature dimension size with respect to the input.

3.3.4.1 Shifted Window Self-Attention

As mentioned in the preceding sub-sections, naively applying self-attention to Fig. 3.10 is infeasible since the memory complexity grows quadratically with respect to the spatial size. To circumvent this issue, Swin Transformer employs Shifted Window Self-Attention where the given feature map is divided into an $M \times M$ grid with self-attention being applied locally to the features lying inside each window. To enable feature interaction across windows, the grid is shifted to the next self-attention layer. Fig. 3.10 illustrates the architecture of two successive Swin Transformer Blocks, denoted by l and $l+1$, along with visualization of the self-attention windows for each block. At first glance, the shifted window approach may come off as strange since the number of windows for $l+1$ (9) is different from that for l (4) and the number of patches in each window is

also different. However, Liu *et al.* (2021) proposed a cyclic shift approach that efficiently resolves this inconsistency. We refer the reader to the full paper for more details.

3.3.4.2 Architecture Variants

As is common practice for backbone architectures, Liu *et al.* (2021) proposed multiple variants of Swin Transformer with different depths and feature dimension sizes. These variants are summarized in Table 3.2.

Table 3.2: Architecture design parameters for various variants of Swin Transformer. L_i refers to the number of Swin Transformer blocks in the i-th stage (see Fig. 3.9 for overall architecture). C is the feature dimension size for the first stage.

Variant	C	L_1	L_2	L_3	L_4
Swin-T(iny)	96	2	2	6	2
Swin-S(mall)	96	2	2	18	2
Swin-B(ase)	128	2	2	18	2
Swin-L(arge)	192	2	2	18	2

3.4 Evaluation Measures

Because video segmentation has ballooned into a major research domain, there are several datasets, benchmarks, and tasks that were proposed by different people from different backgrounds with different motivations. As a result, there are a myriad of evaluation measures that are used to quantify method performance for different tasks. In this section, we discuss some of the more popular measures in terms of their formulation, pros and cons, and the tasks they are applied to.

3.4.1 Mean Average Precision (mAP)

mAP is a common evaluation metric for both video and image-level object detection and segmentation tasks which require objects belonging to a pre-defined set of categories to be detected/segmented. In this case, given C categories, the mAP is computed as the mean of the per-class Average Precision (AP):

$$\texttt{mAP} = \sum_{i=1}^{C} \texttt{AP}_i$$

The per-class AP is in turn calculated as the approximate area under the precision-recall (PR) curve. Each point on the PR curve is obtained by using a certain IoU threshold to compute True Positives, False Positives, True Negatives, and False Negatives for the given set of predicted and ground-truth object masks. For segmentation tasks, it is common to sample the IoU thresholds from the range $[0.5, 0.95]$ with an interval of 0.05.

mAP is the primary evaluation measure for datasets such as YouTube-VIS (Yang *et al.*, 2019a) and OVIS (Qi *et al.*, 2022), and also bounding-box level tracking datasets such as TAO (Dave *et al.*, 2020). These benchmarks generally involve in-the-wild video sequences publicly available on the internet, and the underlying idea is that video-level instance segmentation is an extension of image-level instance segmentation where mAP has long been the metric of choice for benchmarks such as COCO (Lin *et al.*, 2014), ADE20k (Zhou *et al.*, 2017) and Mapillary (Neuhold *et al.*, 2017).

Although this sounds intuitive, there are some drawbacks to this approach. Firstly, mAP does not account for track fragmentation. Since only one true positive is allowed per ground-truth track, all other track fragments are penalized as false negatives. Secondly, for category-based tasks such as instance segmentation, mAP allows object masks to be overlapping as long as they have different predicted class labels. As a result, it has nowadays become common for published works to duplicate the same object tracks with multiple class labels since this generally improves performance even though this is practically nonsensical.

3.4.2 Multi-Object Tracking and Segmentation Accuracy (MOTSA)

MOTSA was proposed by Voigtlaender *et al.* (2019b) as a measure for evaluating object segmentation and tracking quality in video. It is an extension of MOTA (Multi-object Tracking Accuracy) (Stiefelhagen *et al.*, 2007) which is widely used for bounding-box level multi-object tracking. MOTSA is computed as follows:

$$\text{MOTSA} = 1 - \frac{|FN| + |FP| + |IDS}{|M|} = \frac{|TP| - |FP| - |IDS|}{|M|} \tag{3.6}$$

where $|TP|$, $|FP|$, and $|FN|$ denote True Positives, False Positives, and False Negatives, respectively. $|IDS|$ and $|M|$ denote the number of ID switches and the number of per-frame ground-truth object masks, respectively. There is also a 'soft' version of MOTSA, called 'sMOTSA', which is computed by replacing $|TP|$ in Eq. 3.6 with $\hat{TP}$, which is the soft number of true positive predictions that have an IoU of more than 0.5 with the ground-truth mask.

Whereas mAP for video segmentation draws inspiration from the image-level instance segmentation community, (s)MOTSA is heavily inspired by the multi-object tracking

(MOT) community. As a result, we see explicit modeling for tracking-specific phenomena such as ID switches in Eq. 3.6. (s)MOTSA is now the measure of choice for driving scene datasets such as KITTI-MOTS and MOTSChallenge (Voigtlaender *et al.*, 2019b).

However, despite being more tracking-oriented, (s)MOTSA suffers from two major drawbacks: (1) it is not a 'proper metric' since it can span the range $(-\infty, 1)$, and (2) it is well-known that the measure is heavily dependent on the quality of per-frame object segmentation. In other words, the (s)MOTSA score can be significantly boosted by simply improving the quality of the per-frame object masks, regardless of the temporal association quality.

3.4.3 $\mathcal{J}\&\mathcal{F}$

$\mathcal{J}\&\mathcal{F}$ is normally used to report performance for Video Object Segmentation (VOS) benchmarks such as DAVIS (Pont-Tuset *et al.*, 2017) and YouTube-VOS (Xu *et al.*, 2018b) where the task is to segment and track a certain set of objects for which the ground-truth masks are given for the first frame in which each object occurs. Note that this is in contrast to the mAP and (s)MOTSA measures discussed earlier which are normally used for benchmarks involving category-based segmentation. $\mathcal{J}\&\mathcal{F}$ is computed as the arithmetic mean of two metrics:

- $\mathcal{J}$-measure: The Jaccard Index, or Intersection-over-union (IoU) between the predicted and ground-truth masks for each target object. Intuitively, Pont-Tuset *et al.* (2016) use this as a region similarity metric.
- $\mathcal{F}$-measure: This is a contour accuracy metric that evaluates how well the mask contours for the predicted masks align with the ground truth. It is computed by computing the precision P_f and recall R_f by performing bipartite matching between the boundary pixels for ground truth and predicted masks. The $\mathcal{F}$-measure is then computed as $\mathcal{F} = \frac{2P_f R_f}{P_f + R_f}$.

3.4.4 Higher Order Tracking Accuracy (HOTA)

Luiten *et al.* (2020) proposed HOTA as a metric that can be applied to a variety of tasks related to object segmentation in video. This metric tries to mitigate the issue of MOTSA being too dependent on the per-frame detection accuracy. HOTA is computed as follows:

$$\text{HOTA} = \sum_{\alpha} \text{HOTA}_\alpha \qquad\qquad \text{HOTA}_\alpha = \sqrt{\text{DetA}_\alpha \cdot \text{AssA}_\alpha}$$

HOTA is thus computed as the mean of the HOTA_α components for each localization threshold α. HOTA_α is in turn computed as the geometric mean of two terms: the Detection Accuracy (DetA_α) and the Association Accuracy (AssA_α).

The idea is to assign equal weight to the per-frame detection/segmentation quality (DetA) and the temporal association quality (AssA). DetA is computed as follows:

$$\text{DetA}_\alpha = \frac{|\text{TP}|}{|\text{TP}| + |\text{FN}| + |\text{FP}|}$$

where TP, FN, and FP denote True Positive, False Negative, and False Positive detections, respectively. The Association Accuracy, on the other hand, is computed as follows:

$$\text{AssA}_\alpha = \frac{1}{|\text{TP}|} \sum_{c \in \{\text{TP}\}} \mathcal{A}(c) \qquad \mathcal{A}_c = \frac{|\text{TPA}(c)|}{|\text{TPA}(c)| + |\text{FNA}(c)| + |\text{FPA}(c)|}$$

Here, TPA, FPA, and FNA denote True Positive Associations, False Positive Associations, and False Negative Associations, respectively. Symbols in the format $X(c)$ mean that the measure X is being computed for a given true positive detection c. AssA is thus computed by averaging over the $\mathcal{A}_c$ measures across all ground-truth detections.

To justify this metric, Luiten *et al.* (2020) conducted several experiments comparing HOTA to existing measures for various tracking scenarios. Moreover, a user study was also conducted which showed that HOTA correlates better with human-perceived visual tracking quality compared to existing metrics such as sMOTSA, mAP, and IDF1 (Ristani *et al.*, 2016).

4

STEm-Seg: Spatio-temporal Embeddings for Object Segmentation in Video

4.1 Introduction

The task of segmenting and tracking multiple objects in videos is becoming increasingly popular due to a surge in the development of autonomous vehicles and robots that are able to perceive and accurately track surrounding objects. These advances are driven by the recent emergence of new datasets (Caelles *et al.*, 2019; Voigtlaender *et al.*, 2019b; Yang *et al.*, 2019a) containing videos with dense, per-pixel annotations of object instances. The underlying task tackled in these datasets can be summarized as follows: given an input video containing multiple objects, each pixel has to be uniquely assigned to a specific object instance or to the background (see Figure 4.1).[1]

State-of-the-art methods tackling these tasks (Voigtlaender *et al.*, 2019b; Wang *et al.*, 2019a; Yang *et al.*, 2019a) usually operate in a *top-down* fashion and follow the *tracking-by-detection* paradigm which is well-established in multi-object tracking (MOT) (Butt and Collins, 2013; Huang *et al.*, 2008; Leibe *et al.*, 2008a; Okuma *et al.*, 2004). Such methods usually employ multiple networks to detect objects in individual images (He *et al.*, 2017), associate them over consecutive frames using, e.g., optical flow (Sun *et al.*, 2018), and resolve occlusions using learned appearance models. Even though these approaches yield high-quality results, they involve multiple, separate networks, are computationally demanding, and are not end-to-end trainable.

[1]This chapter is based on work published at the European Conference on Computer Vision (ECCV) (Athar *et al.*, 2020) where I and Sabarinath Mahadevan are joint first authors. The code implementation and experiments on VIS and MOTS were mostly done by me, whereas experiments for Unsupervised VOS on DAVIS 2016 were done by Sabarinath Mahadevan.

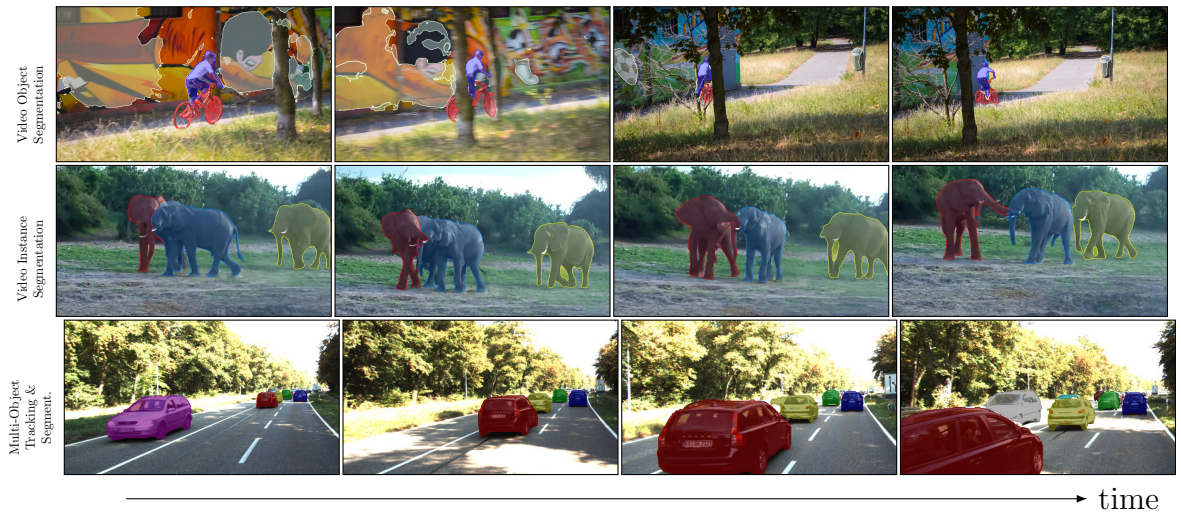


Figure 4.1: Our method is applicable to multi-object segmentation tasks such as VOS *(top)*, VIS *(middle)* and MOTS *(bottom)*.

In this chapter, we propose a different approach. We go back to the roots of vision-based multi-object tracking (Jain and Nagel, 1979; Paragios and Deriche, 2000; Stauffer and Grimson, 2000; Wren *et al.*, 1997) and revisit ideas behind Perceptual Grouping Theory (Palmer, 2002), which postulates that there is evidence for a mechanism in the mammalian vision system that groups similar looking regions into *object instances*. Inspired by this, we leverage the representational power of spatio-temporal embeddings and learn to segment object instances in videos in a *bottom-up* fashion. To this end, we propose an efficient, single-stage network that operates directly on a 3D spatio-temporal volume. We train the embeddings in a category-agnostic setting, such that pixels belonging to the same object instance across the spatio-temporal volume are mapped to a single cluster in the embedding space. This way, we can infer object instances by simply assigning pixels to their respective clusters. Our method outperforms proposal-based methods for tasks involving pixel-precise tracking such as Unsupervised Video Object Segmentation (UVOS) (Caelles *et al.*, 2017, 2019), Video Instance Segmentation (VIS) (Yang *et al.*, 2019a), and Multi-Object Tracking and Segmentation (MOTS) (Voigtlaender *et al.*, 2019b).

To summarize, our contributions are the following: (i) We propose a unified approach for tasks involving instance segmentation in videos (Caelles *et al.*, 2019; Voigtlaender *et al.*, 2019b; Yang *et al.*, 2019a). Our method performs consistently well under highly varied settings such as automotive driving scenes (Voigtlaender *et al.*, 2019b), semantically diverse YouTube videos (Yang *et al.*, 2019a), and scenarios where object classes are not pre-defined (Caelles *et al.*, 2019). (ii) We propose using spatio-temporal embeddings for the aforementioned set of tasks. To this end, we propose a set of mixing functions (Sec. 4.3.2) that improve performance by modifying the feature representa-

tion of these embeddings. Our method enables a simple inference procedure based on clustering within a 3D spatio-temporal volume, thus alleviating the need for external components for temporal association. (iii) We propose a single-stage network architecture that is able to effectively incorporate temporal context and learn the spatio-temporal embeddings. (iv) Code and pre-trained models are publicly available at: https://github.com/sabarim/STEm-Seg.

4.2 Related Work

4.2.1 Image-level Instance Segmentation

Several existing methods for image-level instance segmentation, which is closely related to our task, operate in a *top-down* fashion by using a Region Proposal Network to generate object proposals (Chen *et al.*, 2019; Pinheiro *et al.*, 2015, 2016; Ren *et al.*, 2015), and then performing classification and pixel-precise segmentation (He *et al.*, 2017). Such object proposals are common in tracking (Dave *et al.*, 2019; Osep *et al.*, 2018, 2020; Zeng *et al.*, 2019a; Zulfikar *et al.*, 2019)

There also exist methods which operate in a *bottom-up* fashion by grouping pixels belonging to the same object instance (De Brabandere *et al.*, 2017,?; Kong and Fowlkes, 2018; Neven *et al.*, 2019; Newell *et al.*, 2017; Novotny *et al.*, 2018). De Brabandere *et al.* (2017; 2017) first perform semantic segmentation followed by a density-based clustering in the learned embedding space. Novotny *et al.* (2018) introduces an embedding mixing function to overcome appearance ambiguity and predict a displacement field from the instance-specific center. This is similar to work on Hough voting-based methods for object detection (Leibe *et al.*, 2008b; Qi *et al.*, 2019). Neven *et al.* (2019) extend De Brabandere *et al.* (2017) by training a network to predict object centers and clustering bandwidths, thus alleviating the need for density-based clustering algorithms (Comaniciu and Meer, 2002; Ester *et al.*, 1996; McInnes *et al.*, 2017)p. This serves as a basis for our work. However, in contrast to our approach, the aforementioned methods are only suitable for image-level instance segmentation.

4.2.2 Video Segmentation

Temporally consistent object segmentation in videos can benefit several other tasks such as action/activity recognition (Gkioxari and Malik, 2015; Hou *et al.*, 2017), video object detection (Feichtenhofer *et al.*, 2017; Kang *et al.*, 2017) and object discovery (Kwak *et al.*, 2015; Osep *et al.*, 2019; Wang *et al.*, 2014; Xiao and Jae Lee, 2016). Several *bottom-up* methods segment moving regions by grouping based on optical flow (Van den Bergh *et al.*, 2013; Xu, 2012) or point trajectories (Brox and Malik, 2010; Ochs and Brox,

2012; Palou and Salembier, 2013). By contrast, our method segments and tracks both moving and static objects. Other methods obtain video object proposals (Feichtenhofer *et al.*, 2017; Gkioxari and Malik, 2015; Hou *et al.*, 2017; Kang *et al.*, 2017) based on cues such as actions (Gkioxari and Malik, 2015; Hou *et al.*, 2017) or image-level objects of interest (Kang *et al.*, 2017; Osep *et al.*, 2020). Feature representations are then learned for these proposals in order to perform classification. By contrast, we propose a single-stage approach that localizes objects with pixel-level precision.

4.2.3 Pixel-precise Tracking of Multiple Objects

Segmentation and tracking of multiple objects has its roots in both robotic perception (Teichman *et al.*, 2011) and vision (Jain and Nagel, 1979; Paragios and Deriche, 2000; Wren *et al.*, 1997). Even though some works exist that extend MOT with pixel-precise masks (Milan *et al.*, 2015; Osep *et al.*, 2018), a much larger set of works can be found in the domain of Video Object Segmentation (VOS), which encompasses multiple tasks related to pixel-precise tracking. In the *semi-supervised* variant of VOS (Caelles *et al.*, 2018), the ground-truth masks of the objects that are meant to be tracked are given for the first frame. State-of-the-art approaches to this task (Caelles *et al.*, 2017; Hu *et al.*, 2017; Oh *et al.*, 2019; Tokmakov *et al.*, 2017; Ventura *et al.*, 2019; Voigtlaender *et al.*, 2019a; Wug Oh *et al.*, 2018; Xu *et al.*, 2018b; Yang *et al.*, 2018; Zeng *et al.*, 2019a) usually involve online fine-tuning on the first frame masks and/or follow the *object proposal generation and mask propagation* approach (Luiten *et al.*, 2018; Zeng *et al.*, 2019a; Zulfikar *et al.*, 2019) introduced in the context of *category-agnostic MOT* (Dave *et al.*, 2019; Osep *et al.*, 2018, 2020). Chen *et al.* (2018) tackle this problem by learning embeddings from the first-frame masks, and then associating pixels in the remaining frames to these learned embeddings.

More relevant to our work is the task of *unsupervised* VOS (Caelles *et al.*, 2019; Pont-Tuset *et al.*, 2016). Here, no ground truth information is provided at test time, and the goal is to segment and track all dominant "foreground" objects in the video clip. Current state-of-the-art methods for this task are either proposal-based (Zeng *et al.*, 2019a; Zulfikar *et al.*, 2019) or focus on foreground-background segmentation (Hou *et al.*, 2019; Jain *et al.*, 2017; Siam *et al.*, 2018; Song *et al.*, 2018; Tokmakov *et al.*, 2017; Ventura *et al.*, 2019; Yang *et al.*, 2019b; Yeong Jun Koh, 2017).

Li *et al.* (2018) propose an approach that groups pixel embeddings based on *objectness* and optical flow cues. In contrast to ours, this method processes each frame separately, employs K-means clustering for object localization, and cannot separate different object instances. Wang *et al.* (2019c) employ an attentive graph neural network (Gori *et al.*, 2005) and use differentiable message passing to propagate information across time; we compare our results to theirs in 4.4.5.

Recently, the task of Multi-Object Tracking and Segmentation (MOTS) (Voigtlaender *et al.*, 2019b) and Video Instance Segmentation (VIS) (Yang *et al.*, 2019a) were introduced. Voigtlaender *et al.* (2019b) extended the KITTI (Geiger *et al.*, 2012) and MOTChallenge (Leal-Taixé *et al.*, 2015; Milan *et al.*, 2016) datasets with pixel-precise annotations, and proposed a baseline method that adapts Mask R-CNN (He *et al.*, 2017) to associate object proposals across time. Hu *et al.* (2019) propose an embedding learning method for the MOTS task. Unlike our method, they first use a mask network to filter foreground embeddings and perform mean-shift clustering to generate object masks, which are then associated over time using a distance threshold.

The YouTube-VIS dataset by Yang *et al.* (2019a) contains a large number of YouTube videos with per-pixel annotations for a variety of object classes. Compared to MOTS, these videos typically contain fewer instances but are significantly more diverse. Yang *et al.* (2019a) adapted several existing methods (Bochinski *et al.*, 2017; Han *et al.*, 2016; Voigtlaender *et al.*, 2019a; Wojke *et al.*, 2017a; Yang *et al.*, 2018) for the VIS task, and also proposed their own method (MaskTrack-RCNN) which extends Mask R-CNN with additional cues for data association. We compare these two-stage approaches to our single-stage method in 4.4.2. Other methods such as (Dong *et al.*, 2019; Feng *et al.*, 2019; Liu and Ye, 2019) are either extensions of MaskTrack-RCNN, or use attention mechanism along with additional heuristic post-processing to associate objects over time, unlike our end-to-end bottom-up approach.

4.3 Our Method

We tackle the task of instance segmentation in videos by modeling the video clip as a 3D spatio-temporal volume and using a network to learn an embedding for each pixel in that volume. This network is trained to push pixels belonging to different object instances towards different, non-overlapping clusters in the embedding space. This differs from most existing approaches, which first generate object detections per frame, and then associate them over time. The following sections explain our problem formulation, the network architecture and loss functions employed, and the inference process.

4.3.1 Problem Formulation

As input, we assume a video clip with T frames of resolution $H \times W$. Let us denote the set of RGB pixels in this clip with $\mathcal{X} \in \mathbb{R}^{N \times 3}$ where $N = T \times H \times W$. Assuming there are K object instances in this clip (K is unknown), our aim is to produce a segmentation that assigns each pixel in the clip to either the background or to exactly one of these K instances. We design a network that predicts video instance tubes by clustering pixels simultaneously across space and time based on a learned embedding function.

Instead of learning just the embedding function and then using standard density-based clustering algorithms (Fukunaga and Hostetler, 1975; Lloyd, 1982; McInnes *et al.*, 2017) to obtain instance tubes, we take inspiration from Neven *et al.* (2019) and design a network which estimates the cluster centers as well as their corresponding variances, thus enabling efficient inference. Formally, our network can be viewed as a function that maps the set of pixels $\mathcal{X}$ to three outputs; (1) $\mathcal{E} \in \mathbb{R}^{N \times E}$: an E-dimensional embedding for each pixel, (2) $\mathcal{V} \in \mathbb{R}_{+}^{N \times E}$: a positive variance for each pixel and each embedding dimension, and (3) $\mathcal{H} \in [0,1]^{N}$: an object instance center heat-map.

Instance representation: The network is trained such that the embeddings belonging to the jth instance in the video clip are modeled by a multivariate Gaussian distribution $\mathcal{N}(\boldsymbol{\mu}_j, \boldsymbol{\Sigma}_j)$. Assuming that this instance comprises the set of pixel coordinates $\mathcal{C}_j$ with cardinality N_j, we denote the embeddings and variances output by the network at these coordinates using $\mathcal{E}_j \subset \mathcal{E}$, $\mathcal{E}_j \in \mathbb{R}^{N_j \times E}$ and $\mathcal{V}_j \subset \mathcal{V}$, $\mathcal{V}_j \in \mathbb{R}_{+}^{N_j \times E}$, respectively. During training, $\mathcal{C}_j$ (i.e., the ground truth mask tube for instance j) is known. Using it, the mean $\boldsymbol{\mu}_j$ and covariance $\boldsymbol{\Sigma}_j$ of the distribution are computed by averaging over the per-pixel outputs:

$$\boldsymbol{\mu}_j = \frac{1}{N_j} \sum_{\boldsymbol{e} \in \mathcal{E}_j} \boldsymbol{e} \in \mathbb{R}^E, \qquad \boldsymbol{\Sigma}_j = \frac{1}{N_j} \operatorname{diag}\left(\sum_{\boldsymbol{v} \in \mathcal{V}_j} \boldsymbol{v} \right) \in \mathbb{R}^{E \times E} \ . \tag{4.1}$$

This single distribution models all embeddings belonging to instance j across the entire clip (not individually for each frame). We can now use the distribution $\mathcal{N}(\boldsymbol{\mu}_j, \boldsymbol{\Sigma}_j)$ to compute the probability p_{ij} of any embedding $\boldsymbol{e}_i \in \mathcal{E}$, anywhere in the input clip, of belonging to instance j:

$$p_{ij} = \frac{1}{(2\pi)^{\frac{E}{2}} |\boldsymbol{\Sigma_j}|^{\frac{1}{2}}} \exp\left(-\frac{1}{2} (\boldsymbol{e}_i - \boldsymbol{\mu}_j)^T \boldsymbol{\Sigma}_j^{-1} (\boldsymbol{e}_i - \boldsymbol{\mu}_j) \right) \ . \tag{4.2}$$

Using 4.2 to compute p_{ij}, $\forall\, i \in \{1, .., N\}$, we can obtain the set of pixels $\widehat{\mathcal{C}}_j$ comprising the predicted mask tube for instance j by thresholding the probabilities at 0.5:

$$\widehat{\mathcal{C}}_j = \{ (x_i, y_i, t_i) \,|\, i \in \{1, .., N\},\ p_{ij} > 0.5 \} \ . \tag{4.3}$$

Training: This way, the training objective can be formulated as one of learning the optimal parameters $\boldsymbol{\mu}_j^{\text{opt}}$ and $\boldsymbol{\Sigma}_j^{\text{opt}}$ which maximize the intersection-over-union (IoU) between the predicted and ground-truth mask tubes in the clip:

$$\boldsymbol{\mu}_j^{\text{opt}}, \boldsymbol{\Sigma}_j^{\text{opt}} = \underset{\boldsymbol{\mu}_j, \boldsymbol{\Sigma}_j}{\operatorname{argmax}} \ \frac{\widehat{\mathcal{C}}_j \cap \mathcal{C}_j}{\widehat{\mathcal{C}}_j \cup \mathcal{C}_j}, \tag{4.4}$$

that is, all pixels in the ground truth mask tube $\mathcal{C}_j$ should have a probability larger than 0.5, and vice versa. A classification loss such as cross-entropy could be used to optimize the pixel probabilities. However, we employ the Lovàsz hinge loss (Berman and Blaschko, 2018; Berman *et al.*, 2018; Neven *et al.*, 2019; Yu and Blaschko, 2015) (details in supplementary material), which is a differentiable, convex surrogate of the Jaccard index that directly optimizes the IoU. This formulation allows $\boldsymbol{\mu}_j$ and $\boldsymbol{\Sigma}_j$ to be implicitly learned by the network.

Using Eqs. 4.1-4.4, we can define and optimize the distribution for every instance (i.e. $\forall j \in \{1, ..., K\}$). Note that only a single forward pass of the network is required regardless of the number of instances in the clip. This is in contrast to common approaches for Video Object Segmentation, which only process one instance at a time (Oh *et al.*, 2019; Wug Oh *et al.*, 2018).

We further remark that ours is a *bottom-up approach* which detects and tracks objects in a single step, thus mitigating the inherent drawback of *top-down* approaches that often require different networks/cues for single-image object detection and temporal association. A further advantage of our approach is that it can implicitly resolve occlusions, insofar as they occur within the clip.

4.3.2 Embedding Representation

Under the formulation described in Section 4.3.1, the network can learn arbitrary representations for the embedding dimensions. However, it is also possible to fix the representation by using a mixing function $\phi : \mathbb{R}^E \rightarrow \mathbb{R}^E$ that modifies the embeddings $\mathcal{E}$ as follows: $\mathcal{E} \leftarrow \{\, \phi(\boldsymbol{e}), \mid \boldsymbol{e} \in \mathcal{E} \,\}$.

For the task of single-image instance segmentation, Neven *et al.* (2019) used 2D embeddings ($E = 2$) in conjunction with a spatial coordinate mixing function $\phi_{\text{xy}}(\boldsymbol{e}_i) = \boldsymbol{e}_i + [x_i, y_i]$ [2]. With this setting, the embeddings could be interpreted as offsets to the (x, y) coordinates of their respective locations. The network thus learned to cluster the embeddings belonging to a given object towards some object-specific point on the image. It follows that the predicted variances could be interpreted as the network's estimate of the size of the object along the x and y axes. We postulate that the reason for this formulation yielding good results is that the (x, y) coordinates already serve as a good initial feature for instance separation; the network can then enhance this representation by producing offsets which further improve the clustering behavior. Furthermore, this can be done in an end-to-end trainable fashion which allows the network to adjust the clustering parameters, i.e., the Gaussian distribution parameters, for each instance. In general, it has been shown that imparting spatial coordinate information to CNNs can improve performance for a variety of tasks (Liu *et al.*, 2018; Novotny *et al.*, 2018).

[2] This notation denotes element-wise addition between $\boldsymbol{e}_i$ and $[x_i, y_i]$

Compared to single-image segmentation, the task of associating pixels across space and time in a video clip poses additional challenges, such as object movement, camera ego motion, occlusions, appearance/pose changes, and illumination differences. To tackle these challenges we propose (and experimentally validate in Section 4.4.3) the following extensions:

Spatio-temporal coordinating mixing: Since we operate on video clips instead of single images, a logical extension is to use 3D embeddings ($E = 3$) with a spatio-temporal coordinate mixing function $\phi_{\text{xyt}}(\boldsymbol{e}_i) = \boldsymbol{e}_i + [x_i, y_i, t_i]$.

Free dimensions: In addition to the spatial (and temporal) coordinate dimensions, it can be beneficial to include extra dimensions whose representation is left for the network to decide. The motivation here is to improve instance clustering quality by allowing additional degrees of freedom in the embedding space. From here on, we shall refer to these extra embedding dimensions as *free dimensions*. For example, if $E = 4$ with 2 spatial coordinate dimensions and 2 free dimensions, the mixing function is denoted as $\phi_{\text{xyff}}(\boldsymbol{e}_i) = \boldsymbol{e}_i + [x_i, y_i, 0, 0]$.

There is, however, a caveat with free dimensions: since the spatial (and temporal) dimensions already achieve reasonable instance separation, the network may converge to a poor local minimum by producing very large variances for the free dimensions instead of learning a discriminative feature representation. Consequently, the free dimensions may end up offering no useful instance separation during inference. We circumvent this problem at the cost of introducing one extra hyper-parameter by setting the variances for the free dimensions to a fixed value $\boldsymbol{v}_{\text{free}}$. We justify our formulation quantitatively using multiple datasets and different variants of the mixing function $\phi(\cdot)$ in Section 4.4.3.

4.3.3 Inference

Since the ground truth mask tube is not known during inference, it is not possible to obtain $\boldsymbol{\mu}_j$ and $\boldsymbol{\Sigma}_j$ using 4.1. This is where the instance center heat map $\mathcal{H}$ comes into play. For each pixel $\boldsymbol{c}_i = (x_i, y_i, t_i)$, the value $\mathcal{H}(\boldsymbol{c}_i) \in [0, 1]$ in the heat map at this location gives us the probability of the embedding vector $\mathcal{E}(\boldsymbol{c}_i)$ at this location being an instance center. The sequential process of inferring object instances in a video clip is described in the following algorithm:

1. Identify the coordinates of the most likely instance center $\boldsymbol{c}_j = \text{argmax}_i\, \mathcal{H}(\boldsymbol{c}_i)$.
2. Find the corresponding embedding vector $\mathcal{E}(\boldsymbol{c}_j)$ and variances $\mathcal{V}(\boldsymbol{c}_j)$.
3. Using $\boldsymbol{\mu}_j \leftarrow \mathcal{E}(\boldsymbol{c}_j)$ and $\boldsymbol{\Sigma}_j \leftarrow \text{diag}\,(\mathcal{V}(\boldsymbol{c}_j))$, generate the 3D mask tube $\widehat{\mathcal{C}}_j$ for this instance by computing per-pixel probabilities using 4.2, and then thresholding them as in 4.3.

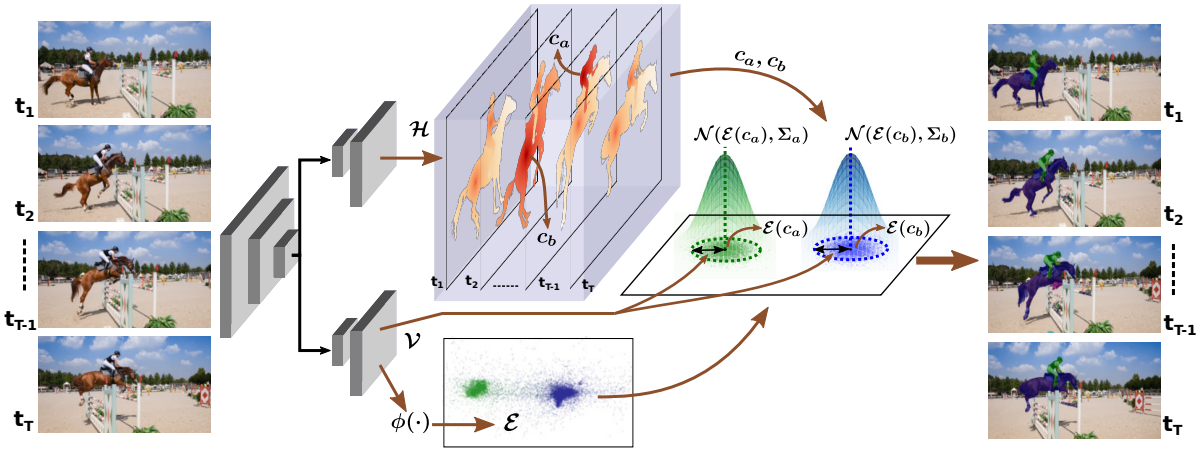


Figure 4.2: Our network (consisting of a backbone with two decoders) takes a video clip as input and produces the embeddings ($\mathcal{E}$), variances ($\mathcal{V}$), and instance center heat map ($\mathcal{H}$). $\mathcal{H}$ contains one peak per object for the entire spatio-temporal volume. In this example, the peaks for the horse and the rider occur at coordinates c_a and c_b, respectively. The corresponding embeddings $\mathcal{E}(c_a)$, $\mathcal{E}(c_b)$ and variances $\mathcal{V}(c_a)$, $\mathcal{V}(c_b)$ at these coordinates are used to define the Gaussian distribution for each object. The probabilities are then thresholded as in Eq. 4.3 to obtain the object mask tubes.

4. Since the pixels in $\widehat{\mathcal{C}}_j$ have now been assigned to an instance, the embeddings, variances and heat map probabilities at these pixel locations are masked out and removed from further consideration:

$$\mathcal{E} \leftarrow \mathcal{E} \setminus \widehat{\mathcal{E}}_j\,, \qquad \mathcal{V} \leftarrow \mathcal{V} \setminus \widehat{\mathcal{V}}_j\,, \qquad \mathcal{H} \leftarrow \mathcal{H} \setminus \widehat{\mathcal{H}}_j. \tag{4.5}$$

5. Repeat steps 1-4 until either $\mathcal{E} = \mathcal{V} = \mathcal{H} = \emptyset$, or the next highest probability in the heat map falls below some threshold.

Even though this final clustering step (Figure 4.2) depends on the number of instances in a clip, in practice, the application of 4.2 and 4.3 carries little computational overhead and its run-time is negligible compared to a forward pass.

Video clip stitching: Due to memory constraints, the clip length that can be input to the network is limited. In order to apply our framework to videos of arbitrary length, we split the input video into clips of length T with an overlap of T_c frames between consecutive clips. Linear assignment (Kuhn and Yaw, 1955) is then used to associate the predicted tracklets in consecutive clips. The cost metric for this assignment is the IoU between tracks in overlapping frames. Our approach is currently *near online* because, given a new frame, the delay until its output becomes available is at most $T - T_c - 1$.

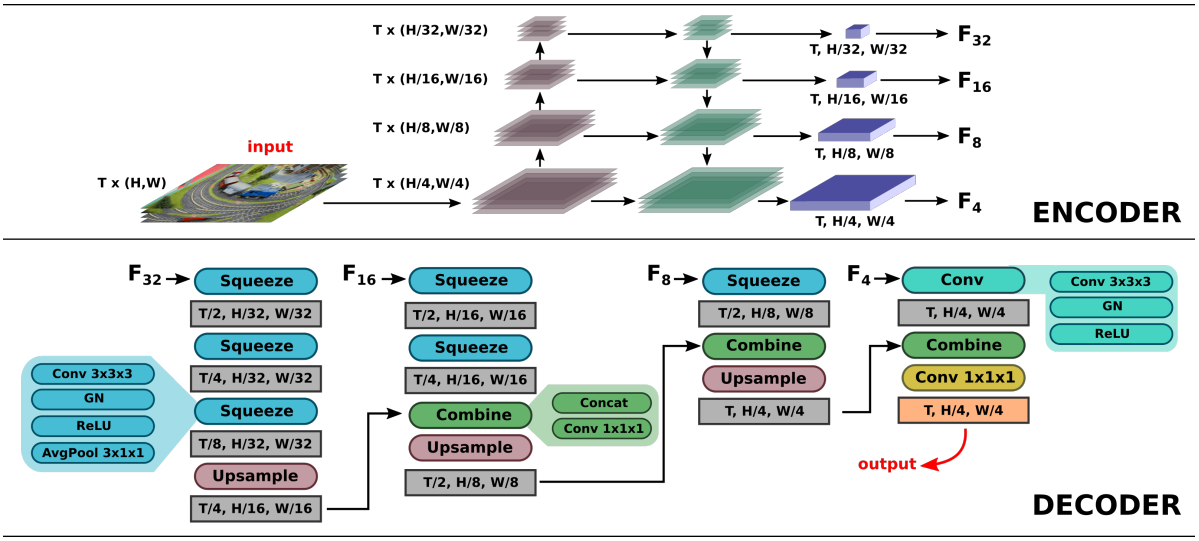


Figure 4.3: The network has an encoder-decoder structure. The encoder (2D backbone + FPN) outputs multi-scale feature maps for each image in the input clip. These feature maps are then stacked along the temporal dimension before being input to decoders with 3D convolutional layers. To reduce clutter, the decoder is only shown once. GN denotes Group Normalization (Wu and He, 2018), and the stride for average pooling is 2x1x1.

4.3.4 Losses

Our model's loss function is a linear combination of three terms:

$$L_{\text{total}} = L_{\text{emb}} + L_{\text{smooth}} + L_{\text{center}} \tag{4.6}$$

Embedding loss L_{emb}: As mentioned in Section 4.3.1, we use the Lovàsz hinge loss to optimize the IoU between the predicted and ground truth masks for a given instance. The embedding loss for the entire input clip is calculated as the mean of the Lovàsz hinge loss for all object instances in that clip.

Variance smoothness loss L_{smooth}: To ensure that the variance values at every pixel belonging to an object are consistent, we employ a smoothness loss L_{smooth} similar to Neven *et al.* (2019). This regresses the variances $\mathcal{V}_j$ for instance j to be close to the average value of all the variances for that instance, i.e. Var$[\mathcal{V}_j]$.

Instance center heat map loss L_{center}: For all pixels belonging to instance j, the corresponding outputs in the sigmoid activated heat map $\mathcal{H}_j$ are trained with an L_2 regression loss to match the output of 4.2. The outputs for background pixels are regressed to 0. During inference, this enables us to sample the highest values from the heat map which corresponds to the peak of the learned Gaussian distributions for the object instances in an input volume, as explained in Sec Section 4.3.3.

4.3.5 Network Architecture

The network (Figure 4.3) consists of an encoder with two decoders. The first decoder outputs the embeddings $\mathcal{E}$ and variances $\mathcal{V}$, while the second outputs the instance center heat map $\mathcal{H}$. The encoder comprises a backbone with a Feature Pyramid Network (FPN) that produces feature maps at 4 different scales for each image in the input clip. The feature maps at each scale are then stacked along the temporal dimension before being input to each of the decoders.

Our decoder consists of 3D convolutions and pooling layers which first compress the feature maps along the temporal dimension before gradually expanding them back to the input size. The underlying idea here is to allow the network to learn temporal context in order to enable pixel association across space and time. To reduce the decoder's memory footprint and run-time, the large-sized feature maps undergo a lower degree of temporal pooling (i.e., fewer convolution/normalization/pooling layers). We call our decoder a *temporal squeeze-expand decoder* (abbreviated as TSE decoder). In Section 4.4.5, we experimentally validate our network's stand-alone ability to learn spatio-temporal context.

4.3.6 Category Prediction

Our task formulation is inherently category-agnostic. However, some datasets (Voigtlaender *et al.*, 2019b; Yang *et al.*, 2019a) require a category label for each predicted object track. For such cases, we add an additional TSE decoder to the network that performs semantic segmentation for all pixels in the input clip, and that is trained using a standard cross-entropy loss. During inference, the logits for all pixels belonging to a given object instance are averaged, and the highest-scoring category label is assigned. Note that this is merely a post-processing step; the instance clustering still happens in a category-agnostic manner.

4.4 Experimental Evaluation

To demonstrate our method's effectiveness and generalization capability, we apply it to three different tasks and datasets involving pixel-precise segmentation and tracking of multiple objects in videos.

4.4.1 Training

For all experiments, we use a ResNet-101 (He *et al.*, 2016) backbone initialized with weights from Mask R-CNN(He *et al.*, 2017) trained for image instance segmentation on COCO (Lin *et al.*, 2014). The temporal squeeze-expand decoders are initialized with

random weights. The network is optimized end-to-end using SGD with momentum 0.9 and an initial learning rate of 10^{-3} which decays exponentially.

Augmented Images. Since the amount of publicly available video data with dense per-pixel annotations is limited, we utilize image instance segmentation datasets by synthesizing training clips from single images. We apply on-the-fly random affine transformations and motion blur and show that this technique is effective (Section 4.4.3) even though such augmented image sequences have little visual resemblance to video clips.

4.4.2 Benchmarks

DAVIS Unsupervised: DAVIS (Caelles *et al.*, 2019) is a popular video object segmentation dataset with 90 videos (60 for training and 30 for validation) containing multiple moving objects of diverse categories. Several DAVIS benchmarks have been introduced over the years; we evaluate our method on the 2019 *Unsupervised Video Object Segmentation* (UVOS) benchmark in which the salient "foreground objects" in each video have to be segmented and tracked. The evaluation measures employed are (i) $\mathcal{J}$-score (the IoU between predicted and ground truth mask tubes), and (ii) $\mathcal{F}$-score (accuracy of predicted mask boundaries against ground truth). The mean of those measures, $\mathcal{J}\&\mathcal{F}$, serves as the final score.

YouTube-VIS: The YouTube Video Instance Segmentation (YT-VIS) (Yang *et al.*, 2019a) dataset contains 2,883 high quality YouTube videos with ∼131k object instances spanning 40 known categories. The task requires all objects belonging to the known category set to be segmented, tracked, and assigned a category label. The evaluation measures used for this task are an adaptation of the Average Precision ($\mathcal{AP}$) and Average Recall ($\mathcal{AR}$) metrics used for image instance segmentation.

KITTI-MOTS: Multi-Object Tracking and Segmentation (MOTS) (Voigtlaender *et al.*, 2019b) extends the KITTI multi-object tracking dataset (Geiger *et al.*, 2012) with pixel-precise instance masks. It contains 21 videos (12 for training and 9 for validation) of driving scenes captured from a moving vehicle. The task here is to segment and track all *car* and *pedestrian* instances in the videos. The evaluation measures used are an extension of the CLEAR MOT measures(Bernardin and Stiefelhagen, 2008) to account for pixel-precise tracking (details in Voigtlaender *et al.* (2019b)). We report these measures separately for each object class.

4.4.3 Ablation Studies

Embedding formulation: We first ablate the impact of using different mixing functions for the embeddings (Section 4.3.2) on the DAVIS'19 `val` dataset with clip length

$T = 8$, and summarize the results in Table 4.2(a). Compared to the identity function baseline ϕ_{identity}, imparting a spatial coordinate offset (ϕ_{xy}) improves the $\mathcal{J}\&\mathcal{F}$ from 57.3% to 61.6%. Adding another embedding dimension with a temporal coordinate offset, as in ϕ_{xyt}, yields a further improvement to 62.6%.

Comparing the mixing function pairs (ϕ_{xyf}, ϕ_{xyt}) where $E = 3$, and (ϕ_{xyff}, ϕ_{xytf}) where $E = 4$, we note that having a free dimension is slightly better than having a temporal dimension since there is a difference of 0.2% $\mathcal{J}\&\mathcal{F}$ for both pairs of functions. This is the case for both DAVIS'19 and YT-VIS,[3] where ϕ_{xyff} yields the best results. For KITTI-MOTS however,[3] temporal and free dimensions yield roughly the same performance. Our intuitive explanation is that for DAVIS and YT-VIS, object instances normally persist throughout the video clip. Therefore, in contrast to the spatial coordinates which serve as a useful feature for instance separation, the temporal coordinate provides no useful separation cue, thus rendering the temporal embedding dimension less effective. On the other hand, using a free dimension enables the network to learn a more discriminative feature representation that can better aid in separating instances. By contrast, objects in KITTI-MOTS driving scenes undergo fast motion and often enter/exit the scene midway through a clip. Thus, the temporal dimension becomes useful for instance separation.

Having additional embedding dimensions is beneficial, but only up to a certain point (i.e., $E = 4$). Beyond that, test-time performance drops, as can be seen by comparing ϕ_{xyff} and ϕ_{xyfff}. We conclude by noting that our proposed formulations for $\phi(\cdot)$ improve performance on video-related tasks compared to existing formulations for single-image tasks (Neven *et al.*, 2019; Novotny *et al.*, 2018). For further discussion and results, we refer to the supplementary material.

Temporal Window Size: Next, we investigate the effect of varying the input clip length on DAVIS'19 `val` (ϕ_{xyff} is used throughout). As shown in Table 4.2(b), larger temporal length helps the TSE decoder to predict better quality mask tubes. Increasing the input clip length from $T = 4$ to $T = 16$ improves the $\mathcal{J}\&\mathcal{F}$ from 62.2% to 64.7%, respectively. Above $T = 24$, the performance decreases.

Training Data: For the DAVIS Unsupervised task, we train on two image datasets, namely COCO (Lin *et al.*, 2014) and Pascal VOC (Everingham *et al.*, 2010), in conjunction with video datasets, namely YT-VIS (Yang *et al.*, 2019a) and DAVIS (Caelles *et al.*, 2019). As shown in Table 4.2(c), this combination yields 64.4% $\mathcal{J}\&\mathcal{F}$ compared to 60.7% when using only video datasets, and 57.1% when using only image datasets. This highlights the benefit of using a combination of image-augmented and video data. For this ablation, $T = 8$ and ϕ_{xyff} were used.

Semantic Head: Since our network does not produce semantic labels for objects, we adapt it to tasks requiring such labels by adding a semantic segmentation decoder as

[3] results for various $\phi(\cdot)$ on YT-VIS and KITTI-MOTS are given in supplementary

Table 4.1: Ablation studies on DAVIS'19 `val`: (a): Impact of different embedding mixing functions; (b): Effect of temporal context; (c): Analysis of training data; (d): Impact of Semantic head on a custom validation split of YT-VIS.

Mixing Function	$\mathcal{J}\&\mathcal{F}$	E
ϕ_{identity}	57.3	2
ϕ_{xy}	61.6	2
ϕ_{xyt}	62.6	3
ϕ_{xyf}	62.8	3
ϕ_{xytf}	64.2	4
ϕ_{xyff}	64.4	4
ϕ_{xyfff}	62.4	5

(a)

Clip Length (T)	$\mathcal{J}\&\mathcal{F}$
4	62.2
8	64.4
16	64.7
24	63.1

(b)

Training Data	$\mathcal{J}\&\mathcal{F}$
Images	57.1
Video	60.7
Images + Video	64.4

(c)

Object Category	$\mathcal{AP}$	$\mathcal{AR}$
Oracle	33.0	34.5
Predicted	24.7	31.8

(d)

explained in Section 4.3.6. Here, we compare the quality of our semantic output to an *oracle* by training our network on a custom train-validation split of YT-VIS. This was done because ground truth annotations for the official validation set are not publicly available. The results presented in Table 4.2(d) show that using *oracle* category labels improves $\mathcal{AP}$ performance by 8.3 from 24.7 to 33.0. This suggests that our results on the official validation set could be further improved by using a better semantic classifier. We leave this for future work.

4.4.4 Comparison with state of the art

Video Object Segmentation: Table 4.3 summarizes our results on the DAVIS'19 unsupervised validation set. `OF-Tracker` and `RI-Tracker` are our own optical flow and re-id baselines which use proposals from a Mask-RCNN (He *et al.*, 2017) network trained with the same backbone and data as our method (see supplementary for details). Our method (64.7% $\mathcal{J}\&\mathcal{F}$) outperforms these baselines and the other published methods by a significant margin, even though we use a single, proposal-free network. AGNN (Wang *et al.*, 2019c), with the second best score of 61.1%, uses object proposals from Mask R-CNN (He *et al.*, 2017) on the salient regions detected by their network and associates them over time using optical flow. We also list the top entries of the DAVIS'19 Challenge Workshop in gray. UnOVOST (Zulfikar *et al.*, 2019) achieves a higher score (67.0%), but

Table 4.3: DAVIS'19 validation results for the unsupervised track. P/D: Proposals/Detections, OF: optical flow, RI: Re-Id, * : uses heuristic post-processing.

DAVIS 2019 Unsupervised											
Method	P/D	OF	RI	$\mathcal{J}\&\mathcal{F}$	$\mathcal{J}$ Mean	$\mathcal{J}$ Recall	$\mathcal{J}$ Decay	$\mathcal{F}$ Mean	$\mathcal{F}$ Recall	$\mathcal{F}$ Decay	fps
KIS* (2019)	✓		✓	59.9	-	-	-	-	-	-	-
UnOVOST* (2019)	✓	✓	✓	67.0	65.6	75.5	0.3	68.4	75.9	3.7	< 1
RVOS (2019)				41.2	36.8	40.2	0.5	45.7	46.4	1.7	20+
AGNN (2019c)	✓	✓		61.1	58.9	65.7	11.7	63.2	67.1	14.3	-
OF-Tracker	✓	✓		54.6	53.4	60.9	-1.3	55.9	63	1.1	∼1
RI-Tracker	✓		✓	56.9	55.5	63.3	2.7	58.2	64.4	6.4	< 1
Ours				**64.7**	**61.5**	**70.4**	**-4**	**67.8**	**75.5**	**1.2**	7

(i) uses several networks along with heuristic-based post-processing, (ii) is an order of magnitude slower (1 vs. 7FPS), and (iii) is highly tailored to this benchmark. To validate this, we adapted UnOVOST to KITTI-MOTS by re-training its networks and optimizing the post-processing parameters with grid search (for further details and analysis of this experiment, we refer to the supplementary). As can be seen in Table 4.5, it does not generalize well to the task of Multi-object Tracking and Segmentation.

Video Instance Segmentation (VIS): This task requires object instances to be segmented, tracked, and also assigned a category label. We therefore adapt our network to this setting as explained in Section 4.3.6. We train jointly on YT-VIS (Yang *et al.*, 2019a) and augmented images from COCO (Lin *et al.*, 2014) (for COCO, we only use the 20 object classes which overlap with the YT-VIS class set). Since this is a new task with few published works, we compare our method to various baselines and adaptions of existing works from Yang *et al.* (2019a) in Table 4.4. As can be seen, our method performs best with respect to all evaluation metrics. Compared to MaskTrack-RCNN (Yang *et al.*, 2019a), we improve the $\mathcal{AP}$ from 30.3 to 34.6, even though MaskTrack-RCNN uses a two-stage object detector and incorporates additional cues during post-processing. Since MaskTrack-RCNN uses a ResNet-50 backbone, we also applied this backbone to our network and still improve the $\mathcal{AP}$ from 30.3 to 30.6.

Multi-Object Tracking and Segmentation (MOTS): Here we again adapt the network for category prediction as required by the task formulation. We outline the results of comparing our method with TrackR-CNN (Voigtlaender *et al.*, 2019b) and UnOVOST (Zulfikar *et al.*, 2019) in Table 4.5. The current top-performing method on MOTS is the two-stage network that extends Mask-RCNN with a re-id head, trained to learn an appearance embedding vector for each object detection, used for data association. Our method achieves the highest sMOTSA score (50.4) on the *pedestrian* class, but TrackR-CNN performs better on the *car* class. However, for both classes, STEm-Seg suffers significantly fewer ID switches (IDS) compared to TrackR-CNN (76 vs. 93 for the

Table 4.4: YouTube-VIS validation results. P/D: Proposals/Detections, FF: First Frame Proposals, OF: Optical Flow.

YouTube Video Instance Segmentation								
Method	FF	P/D	OF	$\updownarrow\mathcal{AP}$	$\mathcal{AP}$@50	$\mathcal{AP}$@75	$\mathcal{AR}$@1	$\mathcal{AR}$@10
`OSMN MaskProp` (2018)	✓			23.4	36.5	25.7	28.9	31.1
`FEELVOS` (2019a)	✓			26.9	42.0	29.7	29.9	33.4
`IoUTracker+` (2019a)		✓		23.6	39.2	25.5	26.2	30.9
`OSMN` (2018)		✓		27.5	45.1	29.1	28.6	33.1
`DeepSORT` (2017a)		✓		26.1	42.9	26.1	27.8	31.3
`MaskTrack R-CNN` (2019a)			✓	30.3	51.1	32.6	31.0	35.5
`SeqTracker` (2019a)				27.5	45.7	28.7	29.7	32.5
`Ours (ResNet-50)`				30.6	50.7	33.5	31.6	37.1
`Ours`				**34.6**	**55.8**	**37.9**	**34.4**	**41.6**

Table 4.5: KITTI MOTS validation set results for Car and Pedestrian class. P/D: Proposals/Detections, OF: optical flow, RI: Re-Id.

KITTI MOTS											
				Car				Pedestrian			
Method	P/D	OF	RI	sMOTSA	MOTSA	MOTSP	IDS	sMOTSA	MOTSA	MOTSP	IDS
`UnOVOST (2019)`	✓	✓	✓	50.7	60.2	85.6	151	33.4	47.7	76.0	68
`MaskRCNN+maskprop (2019b)`	✓	✓		75.1	86.6	87.1	-	45.0	63.5	75.6	-
`TrackRCNN (2019b)`	✓		✓	**76.2**	**87.8**	**87.2**	93	46.8	65.1	75.7	78
`Ours`				72.7	83.8	**87.2**	**76**	**50.4**	**66.1**	**77.7**	**14**

car class and 14 vs. 78 for the *pedestrian* class). This measure is of particular interest to the tracking community since it directly reflects temporal association accuracy.

Similar to UnOVOST (Zulfikar *et al.*, 2019), TrackR-CNN does not generalize well to other tasks. To validate this, we retrained TrackR-CNN on the YT-VIS dataset. However, the resulting $\mathcal{AP}$ was less than 10. We assume this is due to TrackR-CNN that is forced to use a 14×14 ROI-Align layer (He *et al.*, 2017) due to memory constraints. This results in coarse segmentation masks which are heavily penalized by the $\mathcal{AP}$ measure. Furthermore, the ReID-based embeddings can only learn an appearance model, which is a limitation in YT-VIS where similar-looking objects often occur in the same scene.

4.4.5 Segmentation of Salient Regions in Videos

To verify the efficacy of our proposed squeeze-excite decoder architecture under a more generalized setting, we apply it to the DAVIS'16 unsupervised VOS benchmark (Pont-

Table 4.6: Results on DAVIS'16 val. for the unsupervised track. OF: optical flow, CRF: post-processing using CRF.

DAVIS 2016 Unsupervised					
Method	OF	CRF	$\mathcal{J}\&\mathcal{F}$	$\mathcal{J}$-mean	$\mathcal{F}$-mean
LVO (2018)	✓	✓	-	75.9	72.1
PDB (2018)		✓	-	77.2	74.5
MotAdapt (2018)			-	77.2	77.4
3D-CNN (2019)			-	78.3	77.2
FusionSeg (2017)	✓		-	70.7	65.3
LVO (2017)	✓	✓	-	75.9	72.1
ARP (2017)	✓		-	76.2	70.6
PDB (2018)	✓		-	77.2	74.5
AD-Net (2019b)			78.8	79.4	78.2
AGNN (2019c)		✓	79.9	**80.7**	79.1
Ours			**80.6**	80.6	**80.6**

Tuset *et al.*, 2016) and report results. The task here is to predict a binary segmentation mask that captures all the 'salient' regions of the image frames in a given video. Since this task does not require multiple object instances to be individually segmented and tracked, we drop the embedding-based formulation explained in the preceding sub-sections and instead adopt a simplified architecture with just one decoder that predicts the binary segmentation output. This model is trained on randomly sampled clips of 8 frames from the DAVIS and YT-VIS datasets. The results reported in Table 4.6 show that our simplified STEm-Seg setup achieves state-of-the-art results even though it is a compact encoder-decoder architecture whereas existing state-of-the-art approaches are specifically engineered for this task. It is, however, worth noting that not all aspects are directly comparable. For instance, AGNN (Wang *et al.*, 2019c) and AD-NET (Yang *et al.*, 2019b) employ a stronger Deeplabv3 (Chen *et al.*, 2017) network for image feature extraction, whereas for our setup we use additional video training data from YouTube-VIS. Also note that we we do not perform any post-processing, and hence, for a fair comparison, we also report results from other works without computationally intensive post-processing steps.

4.5 Conclusion

In this chapter, we proposed a novel *bottom-up* approach for instance segmentation in videos which models video clips as 3D space-time volumes and then separates object instances by clustering learned embeddings. We enhanced the feature representation of these embeddings using novel mixing functions which yield considerable performance improvements over existing formulations. We applied our method to multiple, diverse

datasets and achieved state-of-the-art results under both category-aware and category-agnostic settings. We further showed that, compared to existing dataset-specific state-of-the-art approaches, ours generalizes much better across different datasets. Finally, we validated our network's temporal context learning ability by performing a separate experiment on video saliency detection and showed that our good results also generalize there.

The publication on which this chapter is based was published in 2020. Our optimistic hope at the time was that STEm-Seg would form the basis for an alternate paradigm to tracking-by-detection where videos would instead be modeled as 3D spatio-temporal volumes. However, it turned out that the time frame of this publication coincided with the emergence of transformer networks for vision-based applications (Carion *et al.*, 2020; Dosovitskiy *et al.*, 2021). This ushered a paradigm shift for object segmentation in video with architectures based on DETR (Carion *et al.*, 2020) achieving significant performance improvement with simplified network architectures based on multi-head attention.

Nonetheless, STEm-Seg remains one of the most well-known and highly cited approaches for Video Instance Segmentation (VIS). Furthermore, follow-up work by Aygun *et al.* (2021) showed that our embedding clustering approach generalized well to the LiDAR domain where they successfully applied to track objects in LiDAR point-cloud sequences. Moreover, our own follow-up work (Mahadevan *et al.*, 2020) showed that 3D convolutions serve as a simple yet effective way of learning temporally consistent image features for dense prediction tasks in video.

5

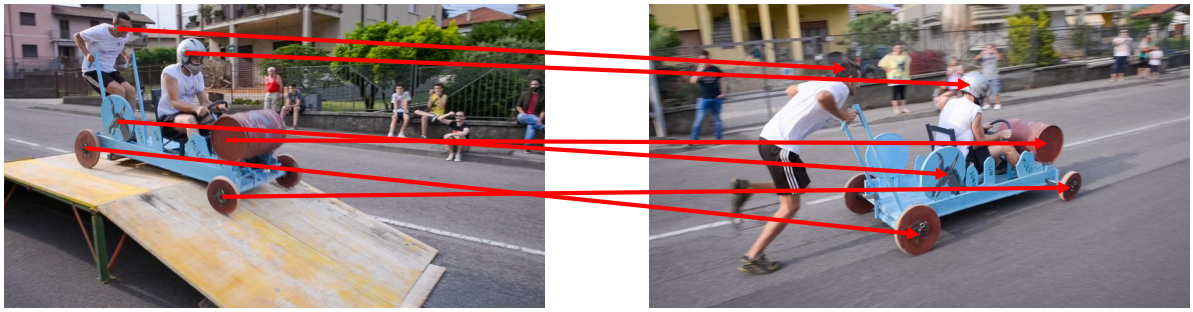
HODOR: High-level Object Descriptors for Object Re-segmentation in Video Learned from Static Images

5.1 Introduction

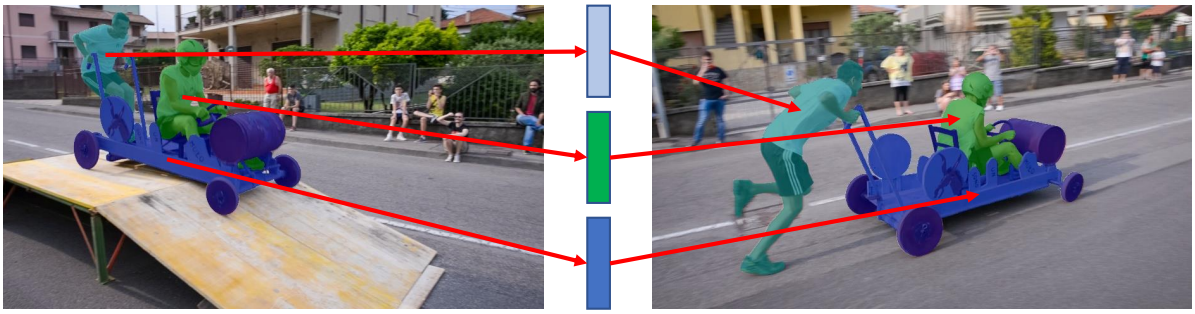
The work in Chapter 4 was focused on developing a state-of-the-art method for object segmentation in video by adopting a different paradigm for processing video. Recall from Sec. 4.4.1 that we pre-trained STEm-Seg on 'fake video' sequences generated by applying random, on-the-fly augmentations to static images from annotated datasets. This approach is widely adopted by the video segmentation community as a means of circumventing the lack of publicly available video data with pixel-precise mask annotations. The work in this chapter focuses on developing a network architecture for Video Object Segmentation (VOS) that is less reliant on video data for training. Note that this task is different to the VIS and MOTS tasks tackled in the previous chapter. For a detailed explanation of how these tasks differ, we refer the reader to Sec. 1.1.2[1].

Current state-of-the-art Video Object Segmentation (VOS) methods learn 'space-time correspondences' (STC), i.e. pixel-to-pixel correspondences, between the image frames in a video. These methods (Cheng *et al.*, 2021c; Oh *et al.*, 2019; Yang *et al.*, 2020) achieve impressive results, but require a large amount of temporally dense annotated video for training. Such datasets require significant human effort, and the annotations are largely redundant since image frames within a video are highly correlated. The largest publicly available VOS dataset (Xu *et al.*, 2018a) contains only a few thousand videos. Single image datasets (Kuznetsova *et al.*, 2020; Lin *et al.*, 2014), in contrast,

[1]This chapter is based on work published at the Conference on Computer Vision and Pattern Recognition (CVPR) (Athar *et al.*, 2022). The code implementation and experiments were done by me. All authors contributed useful ideas and engaged in fruitful discussions.



(a) Space-time correspondence (Cheng *et al.*, 2021b,c; Liang *et al.*, 2020; Lu *et al.*, 2020; Oh *et al.*, 2019; Seong *et al.*, 2020, 2021; Voigtlaender *et al.*, 2019a; Yang *et al.*, 2020, 2021b).



(b) High-level Object Descriptors for Object Re-segmentation (ours).

Figure 5.1: Previous methods (a) learn low-level pixel-pixel correspondence to propagate object masks. HODOR (b) learns high-level object descriptors to re-segment objects in a different frame.

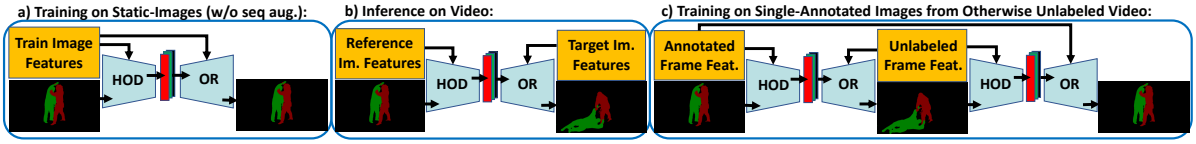


Figure 5.2: HODOR train and inference strategies. HOD: High-level Object Descriptor Encoder. OR: Object Re-segmentation Decoder. Left: HODOR can be trained with single annotated images (without sequence augmentations). Center: HODOR is run on video by feeding features from a different frame to the decoder. Right: Training HODOR can take advantage of unlabeled frames using cycle consistency.

exist with hundreds of thousands of annotated images. In this chapter, we explore the following question: can VOS be learned with only single-image annotations?

To this end, we propose HODOR: **H**igh-level **O**bject **D**escriptors for **O**bject **R**e-segmentation, a novel VOS framework which extracts a robust, *high-level descriptor* for the given objects and background in an image. These descriptors are then used to find and segment those objects in another video frame, i.e. *re-segment* them, even if the object

moves or changes appearance (Fig. 5.1b). This differs fundamentally from STC methods which learn low-level, pixel-to-pixel correspondences (Fig. 5.1a). The underlying idea is that high-level object descriptors can be learned without sequential video data, as this only requires understanding object appearance, and not reasoning about motion. Thus, HODOR can be trained for VOS *using only single images without any video motion augmentation* (Fig 5.2a), and still be applied to video (Fig 5.2b). This is inherently not possible with STC methods since learning correspondences requires comparing multiple, different frames.

The key to our approach is that it forces object appearance information to pass through a concise descriptor, i.e. an information bottleneck. This prevents the descriptor from trivially summarizing the object mask shape and location. The network thus learns to concisely encode object appearance and also to match the descriptor to each pixel in order to re-segment the object in the same image.

If we add sequential augmentation to our single image training strategy to increase the network's robustness, HODOR out-performs all existing methods trained with similar augmented image sequences on the DAVIS (Pont-Tuset *et al.*, 2017) and YouTube-VOS (Xu *et al.*, 2018a) benchmarks. This is because STC methods can only learn correspondences of simple motion from augmented frames, and thus cannot generalize well to the complex motion of real video. HODOR however, being based on high-level object appearance and scene context, is much more resilient to this discrepancy.

HODOR can also be trained using cycle consistency on video where only a single frame is annotated (Fig 5.2c). Without modifying the approach at all, we can simply propagate masks through unlabeled frames and then in reverse back to the labeled frame to apply the loss. This is enabled by a fully differentiable formulation for attending to soft input masks which allows gradients to flow through multiple frame predictions. Based on this, our network can learn to be more robust to appearance changes that occur in natural video, while only requiring single annotated frames. Current STC methods cannot be trained under this setting.

There are two further advantages: The encoder can process, and model interactions between, an arbitrary number of objects. This improves performance and makes the inference speed largely independent of the number of objects. This is in contrast to many works (Cheng *et al.*, 2021b,c; Oh *et al.*, 2019; Yang *et al.*, 2020) where part of the network requires separate forward passes per object. (2) The decoder can jointly attend to object descriptors over multiple past frames with negligible overhead. Thus, we can incorporate temporal history during inference even though the method can be trained on just single images.

To summarize: we propose a novel VOS framework that uses high-level descriptors to propagate objects across video. This enables training using just single images, with or without other unlabeled video frames. Our model processes an arbitrary number of objects simultaneously, and can readily incorporate temporal context during inference.

We achieve state-of-the-art results on DAVIS and YouTube-VOS among methods trained without video annotation. Code and pre-trained models are available at: https://github.com/Ali2500/HODOR

5.2 Related Work

We group existing VOS methods into three categories: pixel-pixel, object-object, and object-pixel. Though not all methods perfectly fit this taxonomy, it is nonetheless useful in comparing our approach to existing works.

Pixel-pixel Correspondence. Such approaches learn low-level space-time correspondence between pixels, and use these correspondences to propagate object masks between video frames. Whereas early VOS approaches (Cheng *et al.*, 2017; Khoreva *et al.*, 2019; Li and Loy, 2018; Perazzi *et al.*, 2017) used pre-computed optical flow as a measure for pixel-pixel correspondence, FEELVOS Voigtlaender *et al.* (2019a) was the first to learn these correspondences in an end-to-end fashion within the VOS framework, and STM (Oh *et al.*, 2019) significantly improved upon this. Nearly all subsequent VOS methods (Cheng *et al.*, 2021b,c; Liang *et al.*, 2020; Lu *et al.*, 2020; Seong *et al.*, 2020, 2021; Yang *et al.*, 2020, 2021b), including the two current state-of-the-arts (STCN (Cheng *et al.*, 2021c) and AOT-L (Yang *et al.*, 2021b)) are based on the space-time correspondence paradigm, with each proposing various novel techniques for improving speed and performance. HODOR diverges from this paradigm by instead learning correspondences between pixels and high-level object descriptors.

Self-supervised Pixel-pixel Correspondence. One set of methods learns pixel-pixel correspondences using unlabelled video via self-supervision. To do this, some methods (Lai *et al.*, 2020; Vondrick *et al.*, 2018) optimize their network with colorization and image reconstruction based training objectives. Other methods (Jabri *et al.*, 2020; Wang *et al.*, 2019d) learn from cyclic consistency by propagating random image patches through a video sequence. HODOR can also be trained with cyclic consistency but with the objective of learning high-level object descriptors rather than low-level pixel correspondences.

Object-object Comparison. Another common VOS approach involves directly comparing object representations (Li and Loy, 2018; Liang *et al.*, 2021; Luiten *et al.*, 2018; Voigtlaender *et al.*, 2020; Zeng *et al.*, 2019a). Such methods first learn object proposals for the target image and then match these proposals to previously tracked objects. This paradigm is inspired by methods in multi-object tracking (Bergmann *et al.*, 2019; Voigtlaender *et al.*, 2019b; Wojke *et al.*, 2017b), and often involves spatial similarity constraints and object ReID vectors (Hermans *et al.*, 2017; Li and Loy, 2018) for tem-

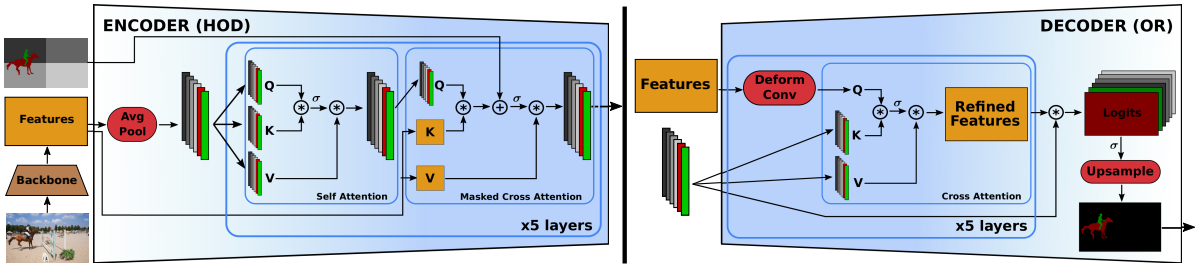


Figure 5.3: The HODOR Architecture consists of a backbone, the HOD encoder, and the OR decoder. Q, K, and V refer to Queries, Keys, and Values, respectively. The encoder jointly encodes all objects and background cells (here 2×2) to descriptors, which are then decoded to masks by the decoder. Some steps are simplified (the final upsampling) or omitted (fc layers, skip connections). See Sec. 5.3 for details.

poral association. Such methods require training for proposal generation on a specific set of object classes and thus do not generalize well to novel categories.

Object-pixel Comparison. An alternative to the pixel-pixel and object-object approaches described above is learning high-level representations for the set of given objects, and then re-segmenting these objects in the target frame by directly comparing the representations to the pixel features in that frame. Early VOS methods followed this paradigm by finetuning a segmentation network during inference on the given first frame object masks (Caelles *et al.*, 2017; Maninis *et al.*, 2018; Perazzi *et al.*, 2017; Voigtlaender and Leibe, 2017) to embed an object representation in the weights of a network, which is then applied directly to subsequent frames. This is extremely slow and usually achieves poor results. The most similar work to ours is SiamMask (Wang *et al.*, 2019b). It learns a vector representation for each object which is directly compared to pixel features to determine whether or not the pixels belong to that object. However, this approach trains on large amounts of annotated video data, and compared to HODOR and other existing methods, does not achieve good results. To the best of our knowledge, no other method from this category even achieves competitive results for VOS.

After online finetuning-based methods fell out of fashion, leaderboards for VOS benchmarks were dominated by object-object association-based methods (Li and Loy, 2018; Luiten *et al.*, 2018) until the emergence of FEELVOS (Voigtlaender *et al.*, 2019a) and STM (Oh *et al.*, 2019). Since then, state-of-the-art VOS approaches have been almost exclusively based on the pixel-pixel correspondence paradigm.

5.3 Method

The HODOR network architecture consists of three components: (1) a backbone that learns multi-scale image features, (2) a High-level Object Descriptor (HOD) encoder,

and (3) an Object Re-segmentation (OR) decoder. With HODOR, we revisit the idea of learning object-level descriptors for VOS which have mostly been replaced in favor of STC-based approaches. To this end, our network architecture enables the essence of an object to be encoded without directly memorizing the object mask's shape or location. We also introduce attention layers which allow multiple objects to be processed simultaneously and allow interactions between their descriptors. These attention layers also enable the descriptors to be enriched with image features (in the encoder), and vice versa (in the decoder).

The architecture is illustrated in Fig. 5.3. Given an RGB image $I \in \mathbb{R}^{H \times W \times 3}$, the backbone produces a pair of C-dimensional feature maps $\mathcal{F} = \{F^4, F^8\}$ at the $4\times$ and $8\times$ downsampled input resolution scales, respectively. Assume that the image I contains O objects of interest with segmentation masks $\mathcal{M}^f = \{M_1^f, ..., M_O^f\}$. We first compute a background mask consisting of all the pixels which do not belong to any object. This background mask is then split into B separate masks $\mathcal{M}^b = \{M_1^b, ..., M_B^b\}$ by dividing it into a grid with B cells.

5.3.1 Encoder

The encoder accepts as input the set of masks $\mathcal{M}^f \cup \mathcal{M}^b$ and the image feature map F^8, and produces a set of object descriptors $\mathcal{D}^f = \{\mathbf{d}_1^f, ..., \mathbf{d}_O^f\}$ containing one C-dimensional descriptor per foreground object, and likewise a set of descriptors $\mathcal{D}^b = \{\mathbf{d}_1^b, ..., \mathbf{d}_B^b\}$ containing one C-dimensional descriptor for each background patch. Intuitively, these descriptors are a concise latent representation of their respective patches (object or background).

Each descriptor is initialized by average pooling the set of pixel features belonging to the corresponding patch. These are then iteratively and jointly refined by a series of transformer-like layers. Each layer consists of multi-head self-attention between the set of descriptors $\mathcal{D}^f \cup \mathcal{D}^b$, followed by multi-head cross-attention in which these descriptors absorb patch-specific information from the feature maps F^8 conditioned on the masks $\mathcal{M}^b \cup \mathcal{M}^f$.

With some abuse of notation, let us use $D^{(l)} = \mathcal{D}^f \cup \mathcal{D}^b \in \mathbb{R}^{(O+B)\times C}$ to denote the set of descriptors at the l-th layer of the encoder and $M = \mathcal{M}^f \cup \mathcal{M}^b \in \mathbb{R}^{(O+B)\times H \times W}$ for the set of patch masks. The l-th layer of our encoder can then be described as follows:

$$\begin{aligned} D^{(l)} &\longleftarrow D^{(l-1)} + \texttt{SelfAttn}(D^{(l-1)}) \\ D^{(l)} &\longleftarrow D^{(l)} + \texttt{MaskedCrossAttn}(D^{(l)}, F^8, M) \\ D^{(l)} &\longleftarrow D^{(l)} + \texttt{FFN}(D^{(l)}) \end{aligned} \tag{5.1}$$

We omit the typical LayerNorm for the sake of text clarity (cf. (Carion *et al.*, 2020)). `FFN` denotes a Feed-forward Network consisting of three fully connected layers with ReLU activations. `SelfAttn` denotes multi-head attention (Vaswani *et al.*, 2017) wherein queries, keys, and values are produced by applying separate linear projections to the input tensor. `MaskedCrossAttn` denotes multi-head attention where queries are produced from the embeddings, but keys and values are produced from the image feature map F^8. We describe this operation as '`Masked`' because we condition the pixel features F^8 on the mask M to enable the descriptors to better focus on their respective patches. This could be done by replacing the dot product affinities between the descriptors D and F^8 in the Key^{T}Query matrix (K^TQ) with $-\infty$ for pixels where the respective mask value for the given patch is zero. However, this operation is not differentiable, and it restricts each descriptor to attend to only those pixels for which the corresponding patch mask value is one. Furthermore, numerical issues can arise for empty masks during training.

Differentiable Soft Attention-masking. We propose a better formulation that is differentiable, allows the mask M to be non-binary (i.e. have soft values), and affords the network more flexibility to focus on relevant image features. Given the mask $M \in [0, 1]$ and a learnable, positive scalar α, we define the attention operation as follows:

$$\texttt{softmax}\left(\frac{K^TQ + \alpha M}{\sqrt{C}}\right) \cdot V \tag{5.2}$$

This is identical to the standard attention operation proposed by Vaswani *et al.* (2017) except for the term '$+\alpha M$'. In practice, each attention head is assigned a different learnable parameter α which is optimized during training. Thus, different attention heads attend to pixel features conditioned on different magnitudes of masking. This enables the network to learn descriptors that focus on their respective patch features, but that are also able to capture scene information from other parts of the image if this is beneficial for the training objective. This is inspired by Press *et al.* (2021) who used additive offsets in temporal attention in NLP.

The encoder is thus designed to learn descriptors conditioned on object/background masks. This formulation contains an inherent information bottleneck that does not allow the input mask's shape or location from directly 'leaking' into the descriptors. Specifically, in Eq. 5.2, the mask M can only influence the `softmax`$(\cdot)$ term, i.e. the weights with which the Values (V) are summed, but M cannot directly be copied into the attention operation output.

5.3.2 Decoder

Whereas the encoder produces descriptors $\mathcal{D}^f \cup \mathcal{D}^b$ by conditioning the image features $\mathcal{F}$ on patch masks $\mathcal{M}^f \cup \mathcal{M}^b$, the decoder does the opposite: it (re)produces the patch masks $\mathcal{M}^f \cup \mathcal{M}^b$ by conditioning the image features $\mathcal{F}$ on the descriptors $\mathcal{D}^f \cup \mathcal{D}^b$. The

architecture is similar to that of the encoder consisting of a series of transformer-like layers with multi-head attention. However, now the image feature map F^8 is updated iteratively by attending to the descriptors. There are two additional differences: (1) the cross-attention does not involve any masking. (2) The self-attention cannot be used for feature maps with large spatial dimensions due to its quadratic memory complexity, so we instead use a 3×3 deformable convolution (Dai *et al.*, 2017). Since the purpose of self-attention is to enrich pixel features by allowing them to attend to all other pixels, a deformable convolution can be thought of as having a similar effect where a pixel can interact with a set of other pixels at learned offsets. Although recent works (Dai *et al.*, 2021; Zhu *et al.*, 2020) proposed efficient variants of attention for image features, we found that deformable convolutions still require less memory.

Let us denote with $F^{8(l)}$ the feature map at the l-th layer of the decoder, and let us use $D \in \mathbb{R}^{(O+B)\times C}$ to denote the descriptors produced by the encoder. The l-th decoder layer can then be described as (again omitting LayerNorms):

$$\begin{aligned} F^{8(l)} &\longleftarrow F^{8(l-1)} + \texttt{DeformConv}(F^{8(l-1)}) \\ F^{8(l)} &\longleftarrow F^{8(l)} + \texttt{CrossAttn}(F^{8(l)}, D) \\ F^{8(l)} &\longleftarrow F^{8(l)} + \texttt{FFN}(F^{8(l)}) \end{aligned} \tag{5.3}$$

For the `CrossAttn`, a linear projection generates the queries from the feature map $F^{8(l)}$ and the keys and values are two separate linear projections of the descriptors D. We omit the final `FFN` from Fig. 5.3 for space reasons.

The final decoder layer outputs a feature map $F^{8(L)}$, which we bilinearly upsample by a factor of 2 and then add to the image feature map $F^4 \in \mathcal{F}$. We then apply a 3×3 convolution to get $F^{4(L)}$ and at this scale we compute the per-pixel object logits based on the dot product between $F^{4(L)}$ and the descriptors D. The resulting logits are upsampled to the input resolution, before applying a `softmax` over the descriptor dimension, yielding the output masks M. Formally, $M \in \mathbb{R}^{H\times W\times(O+B)}$ is calculated as follows:

$$\begin{aligned} F^{4(L)} &\longleftarrow \texttt{Conv}\left(F^4 + \texttt{upsample2}(F^{8(L)})\right) \\ M &\longleftarrow \texttt{softmax}\left(\texttt{upsample4}(F^{4(L)} \cdot D)\right) \end{aligned} \tag{5.4}$$

5.3.3 Video Object Segmentation

So far we discussed how the decoder can reproduce the patch masks which were input to the encoder. However, since the descriptors encode a robust representation of the objects in an image, the decoder can re-segment them in any image I' where these objects exist. Let us use I_t, $\mathcal{F}_t$, $\mathcal{D}_t$ and $\mathcal{M}_t$ to denote the image frame, feature maps, descriptors, and masks at frame t of a given video clip, respectively. Given the first

frame I_1 of a T-frame clip, and the segmentation masks $\mathcal{M}_1^f$ for O objects in the first frame, we can learn a set of descriptors $\mathcal{D}_1^f \cup \mathcal{D}_1^b$ which encode these objects as well as the background (cf. Sec. 5.3.1). We can then segment these objects in another video frame I_t by simply giving the decoder the feature maps $\mathcal{F}_t$ for that frame and conditioning it on the first-frame descriptors $\mathcal{D}_1^f \cup \mathcal{D}_1^b$.

This strategy, however, would not generalize well to lengthy videos with significant scene changes and where objects intersect and occlude each other. In practice, we therefore propagate the object masks sequentially frame-by-frame: $t : 1 \rightarrow 2 \rightarrow ... \rightarrow T$. At each frame t, the encoder creates an updated set of object descriptors $\mathcal{D}_t^f \cup \mathcal{D}_t^b$ from the masks $\mathcal{M}_{t-1}$ predicted for the previous frame (or the initial input masks when $t-1 = 1$). The decoder then segments the objects in frame t using these updated descriptors.

Temporal History. To mitigate large object appearance and scene changes in the video, existing VOS methods (Cheng *et al.*, 2021c; Oh *et al.*, 2019; Yang *et al.*, 2020) incorporate temporal context from multiple past frames when predicting object masks for the current frame. HODOR also achieves the same feat efficiently: recall that the decoder is conditioned on the set $\mathcal{D}^f \cup \mathcal{D}^b$ which contains a variable number of object/background descriptors. To incorporate temporal history when predicting masks for frame t, we simply take the union of the set of descriptors for the past T_p frames which we want to incorporate, i.e. $\mathcal{D}_{t-T_p}^f \cup \mathcal{D}_{t-T_p}^b \cup ... \cup \mathcal{D}_{t-1}^f \cup \mathcal{D}_{t-1}^b$. In the decoder, feature map F_t^8 will be refined by jointly attending to the set of all descriptors in the T_p frame history. The subsequent dot-product with the descriptors will produce a set of masks $M_t \in \mathbb{R}^{H \times W \times T_p \times (O+B)}$. We temporally aggregate over the time dimension to obtain masks for each of the $O + B$ patches (we use `max` for our method).

This formulation has three advantages: (1) it incurs little computational overhead since we only need the $O + B$ descriptors for each past frame instead of the full feature maps. (2) We can train with only single images, and still incorporate temporal context during inference without any architectural changes. (3) We can segment an arbitrary number of objects with a single forward pass of the network. This is in contrast to several VOS methods (e.g. (Cheng *et al.*, 2021c; Oh *et al.*, 2019)) which require a per-object forward pass for at least part of the network.

5.3.4 Training

HODOR's problem formulation makes it versatile with respect to the type of training data it can utilize. For the basic setting, we only need a static image dataset with annotated object masks. However, annotated image sequences, if available, can also be utilized by simply propagating the object (and background) masks over the given sequence.



Figure 5.4: Qualitative results on the DAVIS 2017 validation set: We omit the given first frame masks and only show results for other frames. Note that fish, pigs, rope, guns, and soapbox carts are not annotated in COCO, which is used to train our model.

Furthermore, our problem formulation enables the sequential propagation of object masks over a video to be end-to-end differentiable, i.e. even if we only supervise the masks predicted for the last frame of a given clip, the error will be backpropagated over the entire temporal sequence to the first frame. This allows HODOR to also be trained on unlabeled frames from videos with arbitrarily sparse and temporally inconsistent object ID annotations. Given a training clip with T frames where only frame $t = 1$ is annotated, we can propagate the given object masks from $t : 1 \rightarrow T$, and then further propagate them in reverse temporal order from $t : T \rightarrow 1$. We can then use the principle of cyclic consistency (Jabri *et al.*, 2020; Wang *et al.*, 2019d) for supervision by supervising the predicted masks for $t = 1$ to be identical to the input masks. The inherent information bottleneck of our method enables it to be trained effectively under this setting without trivially copying the input masks across the sequence.

5.4 Experimental Evaluation

Datasets. We evaluate HODOR on the DAVIS'17 (Pont-Tuset *et al.*, 2017) and YouTube-VOS 2019 (Xu *et al.*, 2018a) benchmarks. The DAVIS dataset comprises 60, 30, and 30 video sequences for training, validation, and testing, respectively. YouTube-VOS is a larger dataset with 3471 videos for training and 507 for validation. For both benchmarks the task is to segment and track an arbitrary number of objects in each video. The ground truth mask for each object is only provided for the first frame in which an object appears. The evaluation measures are the $\mathcal{J}$ score (Jaccard Index), $\mathcal{F}$ score (F_1-score), and the average of the two ($\mathcal{J}\&\mathcal{F}$) is treated as the final measure.

Implementation Details. Our backbone network is the 'Tiny' variant of the Swin transformer (Liu *et al.*, 2021) with Feature Pyramid Network (FPN) and both our encoder and decoder consist of 5 layers. For all training settings involving static images, we use the COCO (Lin *et al.*, 2014) dataset. Whenever training on image sequences,

each sequence contains $T = 3$ frames. We provide between 1 and 4 randomly chosen labeled objects per image/sequence. The encoder and decoder weights are randomly initialized, whereas the backbone is initialized from an off-the-shelf checkpoint trained for object detection (Lin *et al.*, 2014). The model is trained using the AdamW optimizer (Loshchilov and Hutter, 2019) with a batch size of 8 parallelized across 4 Nvidia RTX3090 GPUs. During inference, we use a temporal history comprising 7 past frames. The inference runs at $\sim$17 frames/s on an Nvidia RTX3090, independent of the number of instances. We refer readers to the supplementary material for further implementation details e.g. learning rate schedule, and training time.

5.4.1 Training Data Versatility

Table 5.1 shows results on the DAVIS'17 validation set for HODOR trained under different settings. For comparison, we also report results for STCN (Cheng *et al.*, 2021c), the current state-of-the-art VOS method, whenever applicable. On just single images, HODOR achieves 61.6 $\mathcal{J}\&\mathcal{F}$, which is at the level of state-of-the-art VOS approaches from 2017 (Caelles *et al.*, 2017; Voigtlaender and Leibe, 2017) that use online fine-tuning during inference. STCN inherently requires an image sequence and cannot be trained in this setting. In row 2, we train on image sequences generated by duplicating the same image T times without any augmentation. Whereas existing space-time correspondence-based methods collapse under this setting by learning to trivially copy the input mask, HODOR achieves 69.4 $\mathcal{J}\&\mathcal{F}$. Even though this setting does not provide any extra 'information' to the network compared to row 1, the $\mathcal{J}\&\mathcal{F}$ increases significantly from 61.6 to 69.4. This is because the model experiences noisy input masks due to the sequential propagation involved in this training setting. Thus, during inference, the model can robustly track objects across lengthy videos even if the intermediate frame masks are imprecise, as it has encountered similar masks during training.

In row 3, we train on image sequences generated by applying T random affine transformations to static images. With this setting, our $\mathcal{J}\&\mathcal{F}$ further improves to 77.5. This is because such augmentations coarsely approximate video motion, thus making the learned object descriptors more robust to object appearance and scene changes. Existing VOS methods also ubiquitously train on such augmented image sequences as a pretraining step, however, we outperform them under this setting (77.5 vs. 75.8). Qualitative results of this model can be seen in Fig. 5.4 and an analysis of the object descriptors is given in the supplementary material.

We then explore how effectively HODOR can leverage single-frame annotations that are part of a video sequence. For the next two experiments, we utilize the YouTube-VOS (Xu *et al.*, 2018a) and DAVIS (Pont-Tuset *et al.*, 2017) training sets, but assume that only one frame per video (the middle-most frame) is annotated (we only use 3,531 of the 98,797 available video frame annotations). In row 4, we fine-tune models from

Table 5.1: $\mathcal{J}\&\mathcal{F}$ scores for various training settings on the DAVIS 2017 validation set. The sequence length $T = 3$ in all experiments. CC: Cyclic Consistency.

	Training Setting	HODOR	STCN (2021c)
1	Single image	61.6	-
2	$T\times$ duplicated image (no aug)	69.4	-
3	$T\times$ duplicated image (with aug)	77.5	75.8
4	$T\times$ dupl. video frame (with aug)	79.0	72.8
5	T frames, 1 annotated (with CC)	80.6	-
6	Temporally dense video	81.3	85.4

row 3 by similarly augmenting the selected frames. This further improves the $\mathcal{J}\&\mathcal{F}$ from 77.5 to 79.0. STCN on the other hand performs worse (72.8), likely because of overfitting.

For row 5, we finetune the model from row 3, but this time using cyclic consistency by randomly sampling $T - 1$ unlabeled frames around the single annotated frame in each video. This improves the $\mathcal{J}\&\mathcal{F}$ from 77.5 to 80.6. The fact that this is higher than the 79.0 $\mathcal{J}\&\mathcal{F}$ in row 4 shows that HODOR can effectively learn video motion cues from unlabeled frames. Existing supervised STC methods cannot be trained with this strategy since they lack the information bottleneck needed to prevent the network from trivially copying the input mask and also because they cannot back-propagate gradients through the predicted mask.

Finally, we also train using dense video annotations with full supervision (row 6), which improves the $\mathcal{J}\&\mathcal{F}$ from 77.5 to 81.3. STCN out-performs us under this setting (85.4 $\mathcal{J}\&\mathcal{F}$) because the same information bottleneck which enables us to train on single images and unlabeled frames with cyclic consistency also has the drawback of limiting the network's access to fine-grained video motion cues. By contrast, pixel-to-pixel correspondence methods lack such a bottleneck thus enabling them to better leverage dense video data. Nonetheless, to the best of our knowledge, we are the first to surpass 81 $\mathcal{J}\&\mathcal{F}$ on DAVIS'17 using an approach not based on pixel-to-pixel correspondences. Note also that our approach has better scaling properties since we require only one frame annotation per video as opposed to the dense annotations required by existing methods.

5.4.2 Comparison to State-of-the-art

In Tables 5.2 and 5.3, we report results for existing VOS methods categorized by the type of training data used on the DAVIS (Pont-Tuset *et al.*, 2017) and YouTube-VOS (Xu *et al.*, 2018b) datasets, respectively. Results for HODOR are given for two settings: (1)

Table 5.2: Quantitative results on the DAVIS. UI†: Unlabeled Images, OL: Online Fine-tuning, *: retrained by us.

			DAVIS val 17			DAVIS test-dev 17		
		OL	$\mathcal{J}\&\mathcal{F}$	$\mathcal{J}$	$\mathcal{F}$	$\mathcal{J}\&\mathcal{F}$	$\mathcal{J}$	$\mathcal{F}$
UI†	DINO (2021)		71.4	67.9	74.9	-	-	-
Labeled images	OSVOS (2017)	✓	60.3	61.6	69.1	49.0	45.6	52.5
	OnAVOS (2017)	✓	63.6	61.0	66.1	57.7	54.8	60.5
	OSVOS^S (2018)	✓	68.0	64.7	71.3	57.5	-	-
	STM (5x Mix) (2019)		60.0	57.9	62.1	-	-	-
	DMN+AOA (COCO) (2021)		67.9	65.8	70.0	-	-	-
	KMN (5x Mix) (2020)		68.9	67.1	70.8	-	-	-
	STCN (5x Mix) (2021c)		75.8	73.1	78.6	-	-	-
	CFBI (COCO) (2020) *		57.6	55.8	59.4	54.0	51.7	56.3
	STCN (COCO) (2021c) *		55.0	52.3	57.6	51.7	49.5	54.0
	HODOR (Ours, COCO)		77.5	74.7	80.2	65.0	62.6	67.4
	HODOR (Ours, COCO + CC)		**80.6**	**77.8**	**83.4**	**66.0**	**63.6**	**68.4**
Unlabeled videos	MAST (2020)		65.5	63.3	67.6	-	-	-
	STC-CRW (2020)		67.6	64.8	70.2	-	-	-
	MAMP (2021) (uses optical flow)		**70.4**	**68.7**	**72.0**	-	-	-
Labeled videos	FEELVOS (2019b)		71.5	69.1	74.0	57.8	55.2	60.5
	AFB-URR (2020)		74.6	73.0	76.1	-	-	-
	e-OSVOS (2020)	✓	77.2	74.4	80.0	64.8	60.9	68.6
	STM (2019)		81.8	79.2	84.3	-	-	-
	CFBI (2020)		81.9	79.1	84.6	74.8	71.1	78.5
	EG-VOS (2020)		82.8	80.2	85.2	-	-	-
	KMN (2020)		82.8	80.0	85.6	77.2	74.1	80.3
	DMN+AOA (2021)		84.5	81.4	87.5	78.3	74.8	81.7
	HMMN (2021)		84.7	81.9	87.5	78.6	74.7	82.5
	STCN (2021c)		**85.4**	82.2	**88.6**	76.5	73.1	79.6
	AOT-L (2021b)		**85.4**	**82.4**	88.4	**81.2**	**77.3**	**85.1**

when trained on augmented image sequences from COCO (Lin *et al.*, 2014) (cf. Table 5.1, row 3), and (2) after fine-tuning with cyclic consistency using just one labeled frame per training set video (cf. Table 5.1, row 5). We use the same model checkpoint for all three benchmarks. For the sake of completeness, we also list results for methods that do not require any annotations, and also those trained on densely annotated video.

Looking at the 'Labeled Images' category, it can be seen that HODOR trained on COCO achieves 77.5 $\mathcal{J}\&\mathcal{F}$ on DAVIS'17, outperforming all existing methods. This includes earlier VOS methods (Caelles *et al.*, 2017; Maninis *et al.*, 2018; Perazzi *et al.*, 2017; Voigtlaender and Leibe, 2017) that perform online fine-tuning (best score: 68.0 $\mathcal{J}\&\mathcal{F}$ from OSVOS^S (Maninis *et al.*, 2018)), but also current state-of-the-art methods which pre-train on similar augmented image sequences. The best-performing method among these is STCN (75.8 $\mathcal{J}\&\mathcal{F}$) which is 1.7 $\mathcal{J}\&\mathcal{F}$ lower than our 77.5. It is worth

Table 5.3: Quantitative results on YouTube-VOS. We focus on the 2019 validation set, but substitute 2018 validation set results when only those are available (slightly higher, highlighted in grey). As is common, we evaluate unseen (us) and seen (s) object classes separately. OL: Online Fine-tuning, *: retrained by us.

			YouTube-VOS val 18 /val 19				
		OL	$\mathcal{J}\&\mathcal{F}$	$\mathcal{J}_{us}$	$\mathcal{F}_{us}$	$\mathcal{J}_{s}$	$\mathcal{F}_{s}$
Labeled Images	STM (5x Mix) (2019)		69.1	-	-	-	-
	CFBI (COCO) (2020) *		68.4	64.7	70.9	68.2	69.7
	STCN (COCO) (2021c) *		69.4	66.4	73.9	67.8	69.3
	HODOR (Ours, COCO)		71.7	**65.5**	**71.4**	73.7	76.0
	HODOR (Ours, COCO + CC)		**72.4**	63.9	70.5	**75.8**	**79.3**
Labeled videos	AFB-URR (2020)		79.6	74.1	82.6	78.8	83.1
	e-OSVOS (2020)	✓	71.4	74.3	74.3	71.7	66.0
	STM (2019)		79.4	72.8	80.9	79.7	84.2
	CFBI (2020)		81.0	75.2	83.0	80.6	85.1
	KMN (2020)		81.4	75.3	83.3	81.4	85.6
	DMN+AOA (2021)		82.7	76.7	84.8	82.6	87.0
	HMMN (2021)		82.5	77.3	85.0	81.7	86.1
	STCN (2021c)		82.7	78.2	85.9	81.1	85.4
	AOT-L (2021b)		**84.5**	**78.4**	**86.7**	**84.0**	**88.8**

noting that while DMN+AOA (Liang *et al.*, 2021) use COCO images for this training step, STM (Oh *et al.*, 2019), KMN (Seong *et al.*, 2020) and STCN (Cheng *et al.*, 2021c) use a collection of 5 image datasets (Cheng *et al.*, 2020b; Li *et al.*, 2020; Shi *et al.*, 2015; Wang *et al.*, 2017; Zeng *et al.*, 2019b) ('5x Mix' in the table). To verify that this discrepancy does not disadvantage other methods, we retrained STCN and CFBI on COCO images using their respective training code. These experiments are marked with '$*$' in the table. It can be seen that STCN performs significantly worse on DAVIS under this setting (55.0 $\mathcal{J}\&\mathcal{F}$). Though performance on YouTube-VOS is comparatively better (69.4 $\mathcal{J}\&\mathcal{F}$), HODOR still outperforms it (71.7 $\mathcal{J}\&\mathcal{F}$). The same trend holds true for CFBI. One possible explanation for the large performance difference of these methods on the two datasets is that the augmentations applied to static images are quite aggressive, which makes objects undergo significant movement across frames. This better approximates YouTube-VOS videos where objects also frequently undergo large motions. By contrast, object motion in videos from DAVIS is comparatively milder, and because these methods learn pixel-to-pixel correspondences, they do not perform well during inference if the nature of object motion is different from what was encountered during training.

Finally, we also report our result after finetuning with cyclic consistency using only the middle-most annotated frame from each video in the YouTube-VOS and DAVIS training

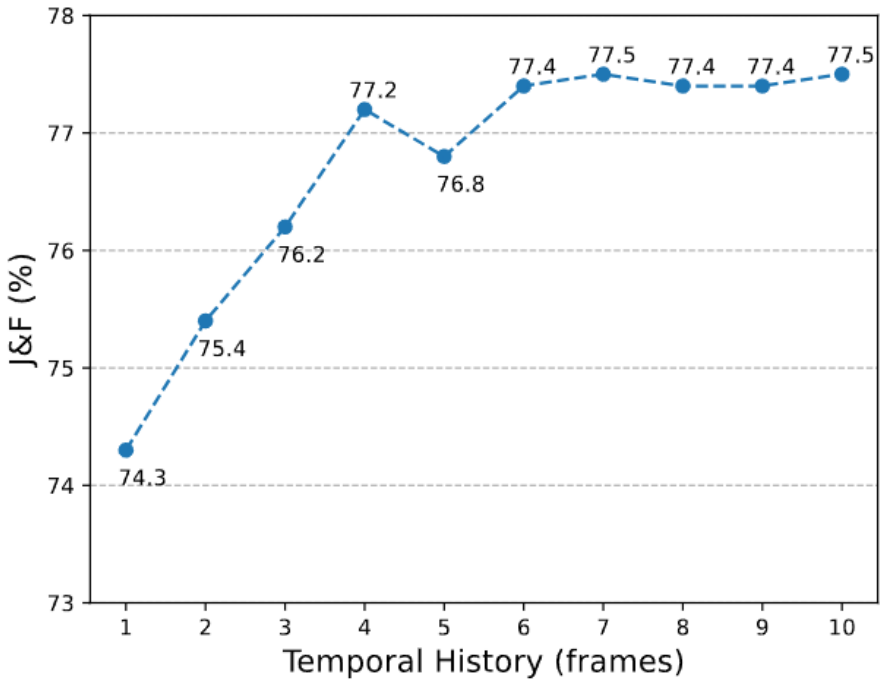


Figure 5.5: Temporal History Ablation: Performance on DAVIS'17 val for different temporal history lengths during inference.

set. This improves the $\mathcal{J}\&\mathcal{F}$ by 3.1, 1.0, and 0.7 points on the DAVIS validation, DAVIS test, and YouTube-VOS validation sets, respectively.

5.4.3 Ablations

We perform ablations to investigate our design choices and report the results in Table 5.4.

Temporal History During Inference. In Sec. 5.3.3, we discussed how HODOR can effectively incorporate temporal history from past frames when predicting the object masks for a given frame. Fig. 5.5 plots the $\mathcal{J}\&\mathcal{F}$ score on DAVIS'17 val for different temporal history lengths. It can be seen that increasing the frame history from 1 to 4 frames yields an approximately linear performance improvement from 74.3 to 77.2. Thereafter, the $\mathcal{J}\&\mathcal{F}$ saturated at 7 frames at a $\mathcal{J}\&\mathcal{F}$ score of 77.5. Recall that because we only need the object/background descriptors for past frames rather than the full feature maps, the inference run-time is minimally affected by the temporal history length: increasing the temporal history from 1 to 10 frames only reduces the inference speed from 17.3 to 16.7 frame/s (reported speed is an average over 5 runs).

Multi-instance. Unlike most other VOS methods, HODOR handles all objects in a single forward pass. This enables it to effectively utilize multi-object context and learn

better descriptors. For experiment (1), we train and infer with only one foreground instance, and merge multiple forward passes by running a pixel-wise `argmax` over the object logits during inference. This reduces the $\mathcal{J}\&\mathcal{F}$ from 77.5 to 71.5, clearly highlighting the benefit of our multi-instance approach which also increases inference speed by negating the need for per-object forward passes.

Attention Masking in Encoder. Recall from Sec. 5.3.1 that we condition the descriptors on their respective patch masks with our proposed soft attention mechanism. Experiment (2) shows the result without attention masking. Thus, the only cue for the descriptors to specialize to their respective targets is their initialization (average pooling over the target pixel features); this reduces the $\mathcal{J}\&\mathcal{F}$ to 74.4. For experiment (3), we apply hard attention masking by thresholding the masks at 0.5 and setting the $\mathrm{K}^T\mathrm{Q}$ matrix entries inside the attention operation to $-\infty$ for pixels where the mask is zero. This strategy yields a similarly reduced $\mathcal{J}\&\mathcal{F}$ of 74.5. The performance increase from using our learned soft attention masking shows that it helps the encoder to better condition the descriptors on the given object/background masks.

Encoder/Decoder Layers. For experiments (4-7), we ablate the depth of the encoder and decoder. Reducing the depth from 5 to 3 in either of them reduces the $\mathcal{J}\&\mathcal{F}$ by $\sim$1. For the zero-layer case, the 5 encoder layers are replaced by a single MLP consisting of 3 fully connected layers, and the 5 decoder layers are replaced with two 3×3 convolutions. For the zero-layer decoder, the $\mathcal{J}\&\mathcal{F}$ reduces to 74.4, whereas the zero-layer encoder reduces the $\mathcal{J}\&\mathcal{F}$ to 72.8. This shows that while both components play an important role in the overall performance, the encoder has a larger impact. One reason could be that whereas self-attention in the encoder allows object descriptors to interact, such interactions are not as profound in the decoder where deformable convolutions are used instead of self-attention.

Barebones Network. For experiment (8), we completely omit both encoder and decoder, i.e. the descriptors are generated by simply average pooling the backbone features, and the output masks are generated by computing their dot product with backbone features of a different image. Doing so reduces the $\mathcal{J}\&\mathcal{F}$ from 77.5 to 70.6. This shows that although the method does not completely collapse without the encoder/decoder layers, the latter is still important and imparts an improvement of 6.9 $\mathcal{J}\&\mathcal{F}$.

Deformable Convolution in Decoder. Due to memory constraints, we use a 3×3 deformable convolution (Dai *et al.*, 2017) instead of the self-attention operation (cf. Sec. 5.3.2). In experiment (9), we instead use a regular 3×3 convolution and observe a reduction in $\mathcal{J}\&\mathcal{F}$ from 77.5 to 75.1. This highlights the importance of substituting self-attention with an operation that is able to attend to far-away spatial locations.

Background Descriptors. We use nine background descriptors initialized by dividing the image into a 3×3 grid and average pooling the background pixel features in each

Table 5.4: Ablation results for various network architecture settings on the DAVIS'17 val set.

	Setting	$\mathcal{J}\&\mathcal{F}$	$\mathcal{J}$	$\mathcal{F}$
1	Single foreground instance	71.5	69.2	73.9
2	No masking in encoder	74.4	71.5	77.2
3	Hard masking in encoder	74.5	71.8	77.1
4	# layers in encoder: $5 \rightarrow 0$	72.8	70.5	75.2
5	# layers in encoder: $5 \rightarrow 3$	76.6	73.9	79.4
6	# layers in decoder: $5 \rightarrow 0$	74.4	71.7	77.1
7	# layers in decoder: $5 \rightarrow 3$	76.4	73.6	79.3
8	Barebones Network	70.6	68.2	72.9
9	Regular convolution in decoder	75.1	72.0	78.2
10	1x background descriptor	76.2	73.7	78.7
	HODOR	77.5	74.7	80.2

cell. This gives HODOR more flexibility to model the background. For experiment (10) we instead use a single background descriptor, reducing the $\mathcal{J}\&\mathcal{F}$ from 77.5 to 76.2.

5.5 Limitations

Aside from our performance on dense video data (discussed in Sec. 5.4.1), another limitation of our method is that when there are distractor objects in the scene with similar appearances, HODOR sometimes compels itself to segment an object even if that object has moved out of the video scene. Since such cases arise more frequently in YouTube-VOS videos, this is one reason why our $\mathcal{J}\&\mathcal{F}$ score for YouTube-VOS is lower than that for DAVIS. In future work, improved training strategies could be formulated to better optimize the model for such challenging cases.

5.6 Conclusion

We proposed a novel VOS approach that uses high-level descriptors for encoding and propagating objects across video. Our approach contains an information bottleneck that enables training on single images and unlabeled frames using cyclic consistency. Thus, unlike existing STC-based methods which train on dense video data, HODOR can be trained on static images, or on videos with arbitrarily sparse, temporally incon-

sistent frame annotations. Since annotating single frames is easier than dense video, HODOR has strong potential for scaling up performance by learning from large-scale video datasets with sparse, or even automatically generated frame annotations (Voigtlaender *et al.*, 2021).

The publication on which this chapter is based was accepted as an 'oral' to the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), which was a status granted to only the top 4.2% of submissions. Since HODOR sacrifices performance for weak supervision capability, it does not feature on state-of-the-art result tables for VOS benchmarks. However, our proposed approach of using an attention-based mechanism to encode objects into concise descriptors has been featured in multiple future works. Rana *et al.* (2023) adopt a similar approach for interactive multi-instance segmentation in images. Tokmakov *et al.* (2023) investigated this work in detail in the context of creating a new VOS benchmark for objects that undergo major appearance-changing transformations. Finally, in our later work (Athar *et al.*, 2023b) which is discussed in Chapter 7, we will see how this mechanism can be easily applied to develop a generalized video segmentation architecture with multi-task capability.

6

BURST: A Benchmark for Unifying Object Recognition, Segmentation, and Tracking in Video

6.1 Introduction

In the preceding two chapters, we discussed two different approaches for tackling object tracking and segmentation in video: (1) STEm-Seg, which performs Video Instance Segmentation (VIS) and Multi-object Tracking and Segmentation (MOTS), and (2) HODOR, which tackles the task of Video Object Segmentation (VOS). At this point, it is understandable if the reader is having a hard time keeping track of these task names and how they differ from each other. These tasks and their associated datasets and benchmarks were developed by different groups of researchers at different points in time with different motivations. Although these benchmarks have facilitated rapid progress in video segmentation, the overall research landscape is now disorganized and fragmented. In this chapter, we propose a new dataset and associated benchmark suite that unifies several tasks related to object segmentation in video under a clear and easy-to-understand hierarchy with shared data and consistent evaluation metrics.[1]

Segmenting and tracking multiple objects in video is widely researched because of applications in autonomous robots and self-driving vehicles. Over time, however, this broadly defined task has splintered into multiple datasets and benchmarks, each with its own sub-community. Even though tasks such as Video Object Segmentation (VOS) and Multi-Object Tracking and Segmentation (MOTS) are closely related, there is a lack of interaction between their sub-communities.

[1]This chapter is based on work published at the Winter Conference on Computer Vision (WACV) (Athar *et al.*, 2023a). The dataset curation, baselines, and experiments were done by me. Jonathon Luiten and Achal Dave managed the annotation process for converting box labels to masks, whereas Paul Voigtlaender wrote the evaluation code. All authors contributed useful ideas and engaged in fruitful discussions.

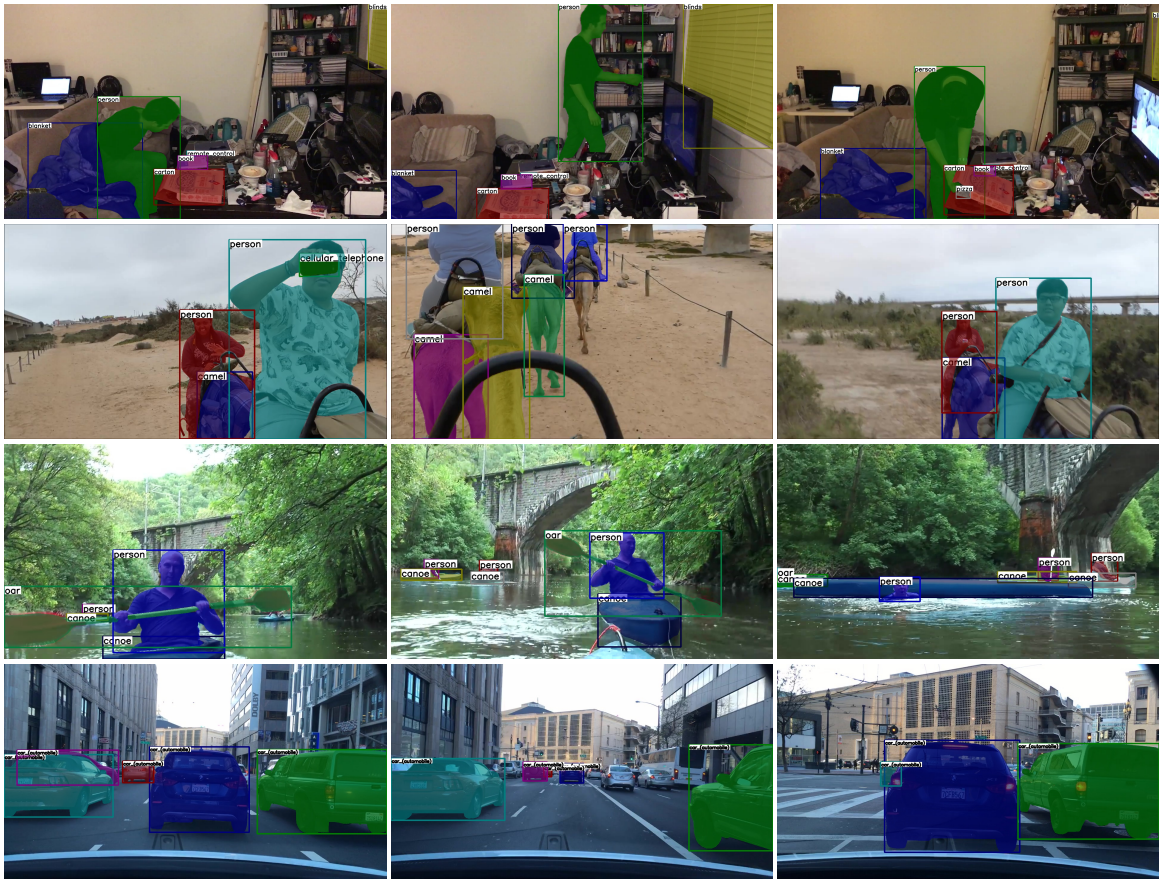


Figure 6.1: Assorted object annotations from BURST showing diverse outdoor, indoor and driving scenes.

The work discussed in this chapter aims to remedy this; we propose BURST: a dataset containing a large, diverse set of videos with object mask annotations, and an associated benchmark with six related tasks. BURST is based on the existing TAO dataset by Dave *et al.* (2020) for bounding-box level multi-object tracking but has been extensively re-annotated with pixel-precise masks. The videos in our dataset include indoor and outdoor scenes, 'in-the-wild' videos, scripted movie scenes, and street scenes captured from moving vehicles. Examples can be seen in Fig. 6.1. The six tasks in our benchmark are organized into a hierarchical taxonomy which is illustrated in Fig. 6.2. All tasks fall under the umbrella of requiring pixel-precise segmentation and tracking of potentially multiple objects in video sequences.

The first level in our task hierarchy splits tasks based on the set of target objects that have to be segmented/tracked. For *exemplar-guided* tasks, an explicit cue is given for each of the target objects. For *class-guided* tasks, the set of target objects is all those that belong to a predefined set of object classes. The exemplar-guided stream is further divided into three tasks where, for the first video frame in which the target object

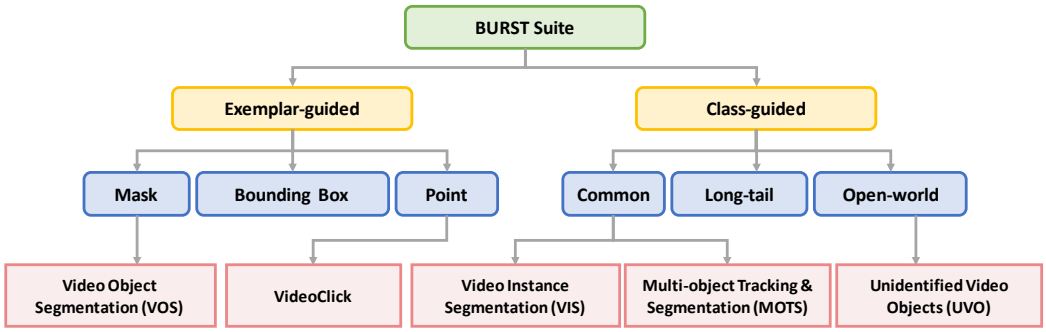


Figure 6.2: BURST Task Taxonomy/Hierarchy. Boxes in the bottom row give examples of existing benchmarks that tackle that task.

appears, we are given either (i) the object mask, (ii) its bounding box or (iii) a random point inside the object. The class-guided stream is also further divided into three tasks where the pre-defined class set is either (i) a small set of common object classes, (ii) a larger set of classes with several infrequently occurring classes (i.e. long-tail), or (iii) an 'open-world' task (Liu *et al.*, 2022) where methods are trained on a small set of known classes, but during inference are expected to additionally track and segment objects belonging to a larger, previously unseen set of classes.

Fig. 6.2 (bottom row) shows which existing benchmarks map onto our task taxonomy e.g. Video Object Segmentation (Pont-Tuset *et al.*, 2017; Xu *et al.*, 2018a) is identical to our mask exemplar-guided task, whereas Video Instance Segmentation (VIS) (Qi *et al.*, 2022; Yang *et al.*, 2019a) and Multi-Object Tracking and Segmentation (MOTS) (Voigtlaender *et al.*, 2019b) are similar to the common class-guided task.

The hierarchy shows that these tasks are highly related to one another. As we will show in Sec. 6.7, research advances targeting one task can be utilized for other tasks. For example, state-of-the-art methods (Cheng *et al.*, 2021c; Oh *et al.*, 2019; Seong *et al.*, 2021; Yang *et al.*, 2021b) for exemplar-guided tasks work by 'propagating' object masks from one video frame to another. Here we note that improvements in mask propagation (i.e. temporal association) can benefit class-guided methods. With BURST, we aim to bring together methods for these tasks under a single umbrella benchmark to encourage more knowledge exchange. To further facilitate unification and interaction, we use the same set of metrics based on Higher-Order Tracking Accuracy (HOTA) (Luiten *et al.*, 2020) for all tasks. This enables direct, quantitative comparison between different methods targeting different tasks. To demonstrate the usefulness of this feature, we set up several effective baselines for our proposed tasks, some of which are constructed by bootstrapping standard approaches for other tasks. The comparability of the resulting scores offers interesting insights into how well methods generalize across tasks.

To summarize, we propose BURST: a large, diverse, and challenging dataset with mask-level object annotations, and an associated benchmark with 6 tasks related to

segmenting and tracking multiple objects in video. Methods can be evaluated for one or more tasks using the same underlying data and comparable metrics. This aims to encourage greater cohesion and knowledge exchange between researchers working on these tasks, and accelerate the development of generalized methods that can tackle multiple tasks. The dataset annotations and parsing API are available at: https://github.com/Ali2500/BURST-benchmark.

6.2 The BURST Benchmark

Existing object tracking and segmentation datasets are typically geared towards certain types of video scenes e.g. in-the-wild internet videos (Pont-Tuset *et al.*, 2017; Qi *et al.*, 2022; Xu *et al.*, 2018a; Yang *et al.*, 2019a), outdoor street scenes captured from a driving vehicle (Chang *et al.*, 2019; Geiger *et al.*, 2012; Homayounfar *et al.*, 2021; Yu *et al.*, 2020). The videos in BURST, on the other hand, cover multiple types of scenes and encompass a large set of 482 object classes. We use the videos from TAO (Dave *et al.*, 2020), which is composed of videos belonging to 7 different datasets: ArgoVerse (Chang *et al.*, 2019) and BDD (Yu *et al.*, 2020), which contain outdoor driving scenes captured from moving vehicles, LaSOT (Fan *et al.*, 2019) and YFCC100M (Thomee *et al.*, 2016), which contain assorted, in-the-wild videos from the internet, and AVA (Gu *et al.*, 2018), Charades (Sigurdsson *et al.*, 2016) and HACS (Zhao *et al.*, 2019), which contain videos with human-human and human-object interactions, but with some subtle differences: Charades contains mostly indoor scenes with slow object motion, AVA contains snippets from scripted movies, and HACS contains in-the-wild internet videos. We refer the reader to Fig. 1 from Dave *et al.* (2020) for an illustrated overview of our dataset classes.

BURST contains 2,914 videos with a lower frame dimension of at least 480px. The videos are ∼30s in length, and the training, validation, and test set contain 500, 993, and 1421 videos, respectively. The training set is annotated at 6fps whereas both validation and test sets are annotated at 1fps. Table 6.1 summarizes statistics for BURST.

Table 6.1: Statistics for BURST train, validation and test sets.

	Train	Validation	Test	Total
Annotation fps	6	1	1	-
Videos	500	993	1421	2,914
Total video length (hrs)	4.94	9.84	14.12	28.9
Object tracks	2,645	5,481	7,963	16,089
Annotated frames	107,144	36,375	52,194	195,713
Object masks	318,200[1]	114,825	167,132	600,157

[1] includes 212,477 automatically generated and consistency-verified masks

Federated Annotations. Similar to TAO (Dave *et al.*, 2020), the annotations in BURST are *federated*, i.e. not all objects belonging to the predefined set of object classes are annotated in every video. This is similar to the philosophy behind the image-level LVIS (Gupta *et al.*, 2019) and OpenImages (Kuznetsova *et al.*, 2020) datasets. Every video in BURST contains the following, in addition to the non-exhaustive annotations: (1) a list of object classes that are non-exhaustively annotated, and (2) a list of object classes that are not present in the video. This information enables us to derive three sets of videos for every object class: where it is present, absent, and non-exhaustively present. This in turn is used to penalize false positives and negatives for each object class during evaluation. We refer readers to the TAO dataset paper (Dave *et al.*, 2020) for more details.

6.3 Comparison to Related Datasets

There exist several datasets of various sizes which tackle one or more tasks evaluated by our benchmark. Tables 6.2 and 6.4 compare BURST to these existing datasets.

6.3.1 Comparison by Tasks

Table 6.4 shows which tasks each dataset/benchmark evaluates in terms of our task taxonomy (Fig. 6.2). We note that existing benchmarks typically address one or at most two tasks. The '/' entries mean that the dataset does not evaluate the given task, but that it is possible to do so e.g. a class-guided benchmark can also be formulated as an exemplar-guided benchmark by assuming that the first-frame object masks are known during inference. For exemplar-guided tasks, the two most common benchmarks are DAVIS (Pont-Tuset *et al.*, 2017) and YouTube-VOS (Xu *et al.*, 2018a). Both contain diverse, in-the-wild, videos from the internet which are ∼5-10s in length. VOT (Kristan *et al.*, 2018) contains longer videos but is a single-object tracking dataset.

Looking at the class-guided stream, most datasets can be assigned to one of two distinct groups. On one hand, benchmarks such as BDD (Yu *et al.*, 2020), KITTI (Geiger *et al.*, 2012) and MOTS-Challenge (Voigtlaender *et al.*, 2019b) are inspired from classical Multi-Object Tracking (MOT). They target autonomous driving problems and contain lengthy videos of street scenes captured from a driving vehicle or a walking pedestrian. We see that KITTI and MOTS-Challenge evaluate methods using 'sMOTSA', which is an extension of the popular MOTA (Multi-Object Tracking Accuracy) measure (Stiefelhagen *et al.*, 2007) for when segmentation masks are used instead of bounding-boxes. On the other hand, datasets like YouTube-VIS (Yang *et al.*, 2019a) and UVO (Wang *et al.*, 2021a) appear more related to Video Object Segmentation (VOS) and typically contain diverse, but shorter videos from the internet. OVIS (Qi *et al.*, 2022) can be seen as an

extension of YouTube-VIS with longer videos and more object occlusion. UVO stands out from the others in that it could be used for the open-world task since it contains mask annotations for anything that humans would consider to be 'objects' as opposed to being restricted to a small class of objects. Datasets in this category use mean Average Precision (mAP) as an evaluation measure. These benchmarks can thus be seen as video extensions of image-level instance segmentation benchmarks such as COCO (Lin *et al.*, 2014), LVIS (Gupta *et al.*, 2019) and OpenImages (Kuznetsova *et al.*, 2020) where mAP is the metric of choice.

In contrast to all of the above, BURST contains the required annotations to evaluate all six tasks. In particular, the long-tail open-world tasks are enabled by the fact that our object class set is sufficiently large.

6.3.2 Comparison by Difficulty

Table 6.2 lists several datasets along with various parameters which subjectively determine their 'difficulty'. In terms of video length, the average sequence in BURST lasts 36.8s, which is longer than the other datasets. Lengthy videos are challenging because of more object instances, and longer occlusions, and are also more memory demanding because of the higher frame count. In terms of the number of annotations, BURST contains a total of ∼600k object masks across ∼200k video frames, which is larger than most other datasets except BDD and UVO. With regard to object classes, BURST contains objects belonging to 482 possible classes, which is significantly higher than the class set for other benchmarks. To the best of our knowledge, we are the first to provide pixel-precise object annotations for such a large set of object classes. Besides increasing object diversity, this feature also enables us to evaluate methods for the long-tail class-guided task. As mentioned in Sec. 6.2 however, our annotations are federated, i.e. not exhaustive. Finally, we note that BURST can better evaluate the generalization capability of methods since it contains more scene diversity compared to existing datasets, many of which focus on specific settings e.g. driving scenes.

6.4 Dataset Creation

We build upon the TAO dataset (Dave *et al.*, 2020) which contains object bounding-box annotations at 1fps. We professionally re-annotated this to obtain pixel-precise masks for all 342,052 object bounding-boxes[2]. We then set out to increase the temporal density of annotations in the training set from 1fps to 6fps. Visualizing a sequence of annotations at 1fps shows large movements and appearance changes between successive frames. It is challenging to train tracking-related methods on this data since they are designed

[2]We re-used the small set of 27,500 masks published by Voigtlaender *et al.* (2021).

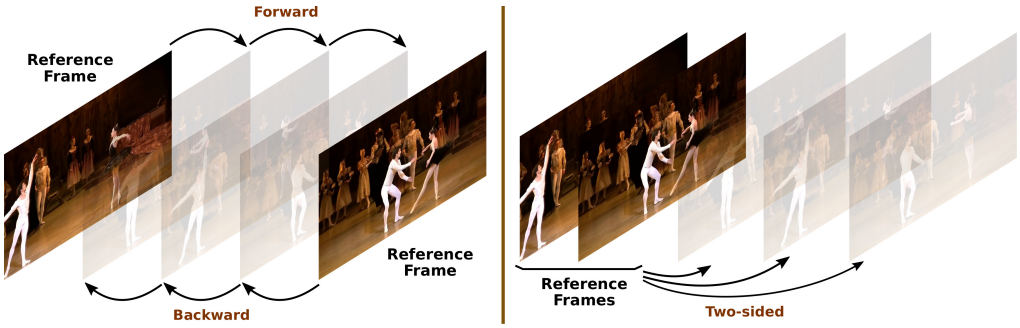


Figure 6.3: Illustration of mask propagation techniques for densifying the training set. STCN (Cheng *et al.*, 2021c) and AOT-L (Yang *et al.*, 2021b) are both executed in the forward and backward settings. The two-sided setting is only executed for STCN.

to learn video motion cues from smooth video frame progression. However, annotating object masks at the full video frame rate (24-30fps) would be in-feasibly expensive and highly redundant since there is usually little scene change between successive frames. Annotating at 6fps is thus a compromise (also used by other datasets (Pont-Tuset *et al.*, 2017; Xu *et al.*, 2018a; Yang *et al.*, 2019a)) since it reduces annotation cost while still ensuring smooth scene progression. For BURST, however, even 6fps annotations would require 255,654 additional mask annotations for the training set. To reduce cost and human effort, we instead developed a semi-automatic procedure to do this, as explained below:

1. Automatic Mask Propagation Interestingly, the task of temporally densifying annotations is practically identical to the mask exemplar guided task referred to in Fig. 6.2: given an object mask in a certain frame, we need masks for the same object in other video frames. In this case, the video length over which this mask propagation step has to be performed is quite short – at most 1s since we already have human-labeled 1fps annotations. We found that two recent, state-of-the-art methods for 'Video Object Segmentation', namely STCN (Cheng *et al.*, 2021c) and AOT-L (Yang *et al.*, 2021b), perform this task quite well with off-the-shelf trained weights. To further improve mask quality, we obtain two different sets of results from each of these methods by running them in two different ways. As shown in Fig. 6.3 (left), given a pair of annotated frames with a number of non-annotated frames in between, we can either run the method with the first frame as the reference and propagate *forward* sequentially, or, starting from the last frame as a reference, propagate *backward* sequentially. Doing so for both methods results in a total of 4 different sets of propagated masks. Additionally, we use STCN to obtain a fifth, tie-breaking *two-sided* result by using both annotated frames as the reference frames (Fig. 6.3, right), and propagating the masks directly to each of the other non-annotated frames (i.e. the frame history update mechanism in STCN is disabled). Thus, we have a total of 5 masks for each object. We subsequently perform a per-pixel majority vote to obtain a final *consensus mask*.

2. Mask Quality Assessment Although most annotations produced by step 1 are high quality, there are several failure cases e.g. poor lighting, occluded scenes, and erratic camera motion. To identify them, one could manually inspect each object mask and decide if it is of ground-truth quality. Even though doing this is still significantly less costly than fully annotating the object masks, we nonetheless developed a more efficient yet effective procedure for evaluating mask quality: we compute the IoU of each of the five masks generated in step 1 with the *consensus mask*. The five resulting IoUs are then averaged to obtain a final metric in $[0, 1]$ which is treated as a quality score Q for the *consensus mask*.

3. Manual Re-annotation of Low-Quality Masks. To decide which masks are of sub-standard quality, we consider two key measures: the score Q from step 2, and the pixel mask area of the consensus mask. We asked two professional annotators to manually assess the quality of a set of 250 object masks. These were sampled such that they were uniformly distributed with respect to their Q scores and pixel areas. The annotators were asked to assign one of three ratings to each object mask: (1) 'good': mask quality is as good as human annotated ground truth, (2) 'satisfactory': there are visible errors e.g. object contours are imperfect, minor instances of mask fragmentation, but overall still acceptable and (3) 'bad': there are unacceptable errors e.g. object ID switches, gross under/over-segmentation. Fig. 6.4 illustrates the results of this survey: each object mask is shown as a point whose color reflects the human-assigned rating. The points are plotted w.r.t their Q score and mask pixel areas. We observe a strong correlation between both of these measures and the human-perceived mask quality since most 'bad' masks are in the lower-left corner of the plot, and vice versa. Based on this plot, we decided to manually re-annotate all object masks whose Q scores were below 0.8, or whose mask areas were smaller than 750 pixels. This region is highlighted in red in Fig. 6.4.

By using this workflow to densify the training set annotations from 1fps to 6fps, we only require 43,177 out of 255,654 object masks (16.9%) to be manually annotated since the remaining automatically generated masks passed the quality threshold to be considered ground truth.

6.5 BURST Task Taxonomy

As explained in Sec. 6.1, several existing benchmarks involve closely related tasks pertaining to segmenting and tracking multiple objects in video, but there is limited cross-interaction between their respective research sub-communities. With BURST, we aim to unify these disparate benchmarks under a single umbrella with shared data and consistent evaluation metrics. The six tasks constituting BURST are illustrated in Fig. 6.2 and explained below.

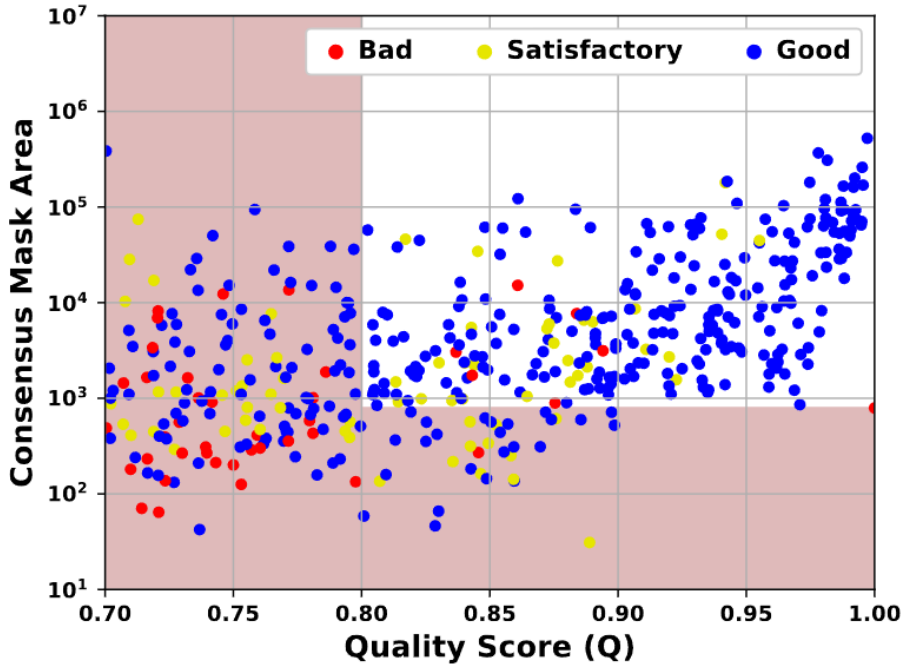


Figure 6.4: Human assessment of mask quality plotted against our automated quality score Q and pixel area of the consensus masks. All masks falling in the red region were manually re-annotated.

6.5.1 Exemplar-guided

This set of tasks requires tracking and segmenting multiple target objects in a video given some ground-truth cue for each of these objects in the first video frame in which they appear. Note that this may not necessarily be the first frame of the video. The three tasks in this stream are based on the type of the given cue:

1) Mask. The method is given the segmentation mask for each of the target objects in the first frame.

2) Box. The bounding box coordinates of the target objects in the first frame are given. Note that the predicted output should still be pixel-precise segmentation masks.

3) Point. This is the most challenging of the three where the method is only given one pixel coordinate which lies inside the target object mask. Again, the predicted output should still contain segmentation masks.

6.5.2 Class-guided

For this set of tasks, methods are required to track, segment, and assign a class label to all objects in a video that belong to a pre-defined set of object classes. The three tasks in this stream are:

4) Common. Here, the target class set includes 78 classes from the popular COCO dataset (Lin *et al.*, 2014) spanning diverse object categories e.g. animals, persons, vehicles, furniture, and food items.

5) Long-tailed. This task involves a large set of 482 object classes from the LVIS dataset (Gupta *et al.*, 2019). It is challenging because several classes contain very few training samples.

6) Open-world. The idea behind open-world instance segmentation (Liu *et al.*, 2022) is that methods are trained on a certain, 'known' set of object classes, but during inference, they are expected to additionally segment objects belonging to an 'unknown' class set. Methods need not assign class labels to the predicted instances, and the evaluation does not penalize false positives. For our open-world task, the 78 'common' classes are the 'known' set, and the 'unknown' set includes everything that is in the 482 class 'long-tail' set, but not in the 'common' set.

6.6 Unified Evaluation Metrics

We evaluate all tasks using Higher Order Tracking Accuracy (HOTA) (Luiten *et al.*, 2020) because it strikes a good balance between measuring frame-level detection and temporal association accuracy. For the open-world task, a slightly modified, recall-based variant of HOTA called Open World Tracking Accuracy (OWTA) is used.

HOTA. To calculate HOTA (Luiten *et al.*, 2020), the predicted detections (per frame) are first matched to the ground truth detections based on the IoU between their masks. Using this mapping, the Detection Accuracy (DetA) and the Association Accuracy (AssA) can be calculated and combined by taking their geometric mean to obtain the HOTA score, i.e.

$$\text{HOTA} = \sqrt{\text{DetA} \cdot \text{AssA}}. \tag{6.1}$$

Multiple IoU thresholds are used to compute the prediction $\leftrightarrow$ ground-truth matching; the final HOTA (and DetA, AssA) is calculated by averaging over the thresholds.

Detection Accuracy (DetA). Using the mapping between predicted and ground truth detections, these detections can be partitioned into a set of True Positive (TP), False Positive (FP), and False Negative (FN) detections. The DetA, which solely mea-

sures the quality of the detections while disregarding track associations, can then be obtained by

$$\text{DetA} = \frac{|\text{TP}|}{|\text{TP}| + |\text{FN}| + |\text{FP}|}. \tag{6.2}$$

Association Accuracy (AssA). To calculate the AssA, an association score $\mathcal{A}(c)$ is calculated for each true positive detection c. The final AssA score is obtained by averaging over the set of true positive detections TP:

$$\text{AssA} = \frac{1}{|\text{TP}|} \sum_{c \in \{\text{TP}\}} \mathcal{A}(c). \tag{6.3}$$

The association score $\mathcal{A}(c)$ for true positive detection c is calculated as

$$\mathcal{A}(c) = \frac{|\text{TPA}(c)|}{|\text{TPA}(c)| + |\text{FNA}(c)| + |\text{FPA}(c)|}, \tag{6.4}$$

where True Positive Associations (TPAs), False Positive Assocations (FPAs), and False Negative Associations (FNAs) (Luiten *et al.*, 2020) are computed by comparing the whole predicted track which goes through detection c with the whole ground truth track which goes through detection c. We refer the reader to Luiten *et al.* (2020) for detailed explanations.

Object Classes. To handle multiple object classes, HOTA can be calculated separately for each class, followed by an averaging step to yield a final metric. To facilitate easier performance analysis over different object classes, we average the per-class HOTA scores over three different sets of object classes: (1) 'common' set, which contains 78 object classes from COCO (Lin *et al.*, 2014), (2) 'uncommon' set, which contains 404 infrequently occurring object classes from LVIS (Gupta *et al.*, 2019), and (3) 'all' set, which is the union of both ($78 + 404 = 482$ classes). We denote these three metrics with HOTA_{com}, HOTA_{unc}, and HOTA_{all}, respectively.

HOTA for Exemplar-guided Tasks. The evaluation for the exemplar-guided task is identical to the class-guided task, and the scores can be directly compared. However, it should be noted that the exemplar-guided methods inherently receive extra ground-truth information: a mask/box/point, and the class label for each target object.

6.6.1 Open-world Evaluation

For the open-world task, methods are expected to segment and track objects of previously unseen classes. Since it is infeasible to label every single object (even the definition of 'object' is ambiguous), we have to assume that the prediction may contain valid objects that are not covered by the ground truth. This entails that false positive detections

should not be penalized, and hence, for the open-world task, we replace HOTA with Open-World Tracking Accuracy (OWTA) (Liu *et al.*, 2022), which is calculated as:

$$\text{OWTA} = \sqrt{\text{DetRe} \cdot \text{AssA}}, \tag{6.5}$$

where the Detection Recall (DetRe) is given by

$$\text{DetRe} = \frac{|\text{TP}|}{|\text{TP}| + |\text{FN}|}, \tag{6.6}$$

Note that DetRe is similar to DetA, but it disregards false positives (FP). To prevent methods from obtaining a high score by simply predicting an extremely large number of detections, we mandate that object mask predictions for the open-world task cannot overlap with each other.

6.7 Baselines

For each of our six tasks (Sec. 6.5), we implement baselines utilizing existing works with off-the-shelf trained models. These serve as a point of comparison for future works and also show how one approach can be utilized for multiple tasks, and how performances across tasks can be directly compared and analyzed. In general, we utilize image-level object detectors in the context of 'tracking-by-detection', where the tracking task is conceptually divided into two steps: a 'detection' step in which objects are segmented in individual frames, followed by a 'tracking' step in which the per-frame detections are associated over time. Although recent state-of-the-art methods (Athar *et al.*, 2020; Cheng *et al.*, 2021a; Mahadevan *et al.*, 2020; Wang *et al.*, 2021c) diverge from this paradigm by jointly segmenting and tracking objects in video clips, we find that it remains a versatile approach for tackling the tasks in BURST. We construct functional baselines for each task using some variation of tracking-by-detection. The following sub-sections detail each of the baselines and the results are presented in Table 6.5.

6.7.1 Exemplar-guided

We show two baselines for each task in this stream: (1) applying STCN (Cheng *et al.*, 2021c), which is a recent 'Video Object Segmentation' method for propagating object masks through video, and (2) a simple box tracker that builds object tracks by starting from the given first-frame mask and then associating object detections in future frames using Hungarian matching based on their bounding-box overlap.

6.7.1.1 Mask-guided

Here, STCN consistently outperforms the box tracker for both class sets since it is a state-of-the-art method for exemplar-guided tracking whereas the box tracker is a basic approach. Note how the difference widens for uncommon classes where STCN achieves 49.2 HOTA_{unc} (validation) whereas box tracker only achieves 13.6. This is because STCN is class-agnostic, and can track any given first-frame object mask, whereas the box tracker uses object detections produced by a Mask2Former (Cheng *et al.*, 2021a) model trained on COCO (Lin *et al.*, 2014) (i.e. 'common' classes). Nonetheless, the applicability of tracking-by-detection for this task shows that exemplar-guided tasks may benefit from future improvements to image-level detectors.

6.7.1.2 Box-guided

or the box and point-guided tasks, we formulate baselines by treating them as extensions of the mask-guided task with an additional 'box $\rightarrow$ mask' or 'point $\rightarrow$ mask' pre-processing step which regresses the segmentation mask from the given first-frame bounding-box or point, respectively. For the box-guided task, we do this in two ways: (1) We compute the IoU between the given bounding box with all the bounding boxes of the image detections for that frame and assign the mask belonging to the detection with the highest overlap, and (2) we input the given first-frame bounding box to a PointRend (Kirillov *et al.*, 2020) based mask regression head from a MaskRCNN (He *et al.*, 2017) model and use the resulting segmentation mask. Looking at Table 6.5, we see that the scores for box-guided are generally lower than those for mask-guided due to the additional 'box $\rightarrow$ mask' regression step. Among box-guided scores, PointRend performs much better than using the best-matched detection. For HOTA_{unc} in particular, the PointRend baseline achieves 44.3 (validation) whereas the matched detection baseline only gets 18.7. Given the fact that both networks (the image detector used for matching, and the PointRend mask head) are trained on COCO, it shows that PointRend is a much more robust 'box $\rightarrow$ mask' regressor than matching detections.

6.7.1.3 Point-guided

Finally, for the point-guided task, we implement 'point $\rightarrow$ mask' by taking the mask for the highest scoring detection which contains the given point. Since this technique is prone to errors, we see that in terms of all three metrics, the baselines for the point-guided task are worse compared to those for box-guided and mask-guided.

6.7.2 Class-guided

We show two tracking-by-detection baselines per task: (1) A simple box tracker which links per-frame object detections using box IoU followed by Hungarian matching, and (2) an 'STCN tracker', where, in order to associate object detection masks in frame t with those in $t+1$, we use STCN to propagate the masks from frame $t+1$ to frame t, and then use the IoU between these propagated masks and the object masks in frame t as an association metric.

6.7.2.1 Common

This task requires segmenting and tracking objects belonging to the 78-class 'common' set. Here, STCN tracker performs better than the box tracker (51.2 vs. 45.5 HOTA_{com}) because STCN-based mask propagation is more accurate for temporal association compared to bounding-box IoU. By effectively utilizing STCN, a method designed to tackle the mask exemplar-guided task, for class-guided tracking, we exemplify the related nature of these tasks and the potential for knowledge exchange among them. Note that we do not evaluate the common task for 'uncommon' classes since this is not required by the task definition.

6.7.2.2 Long-tail

Here, methods are required to segment and track objects belonging to the 482-class 'all' set. We note that the scores here are significantly worse than those for the common task. This is because we used a MaskRCNN (He *et al.*, 2017) model trained on LVIS (Gupta *et al.*, 2019) to obtain object detections for this larger set of classes. We observed that the detections produced by this network are of poor quality. Even for the 'common' classes, the performance of both baselines reduces drastically when these detections are used (27.0 HOTA_{com} on validation for box tracker vs. 45.5 for the common task). Also note that here, the STCN tracker performs worse than the box tracker, even though the opposite was true for the common task. The reason is that STCN performs erroneous mask propagation when the input mask quality is bad. Hence, in this case, the more basic bounding-box IoU tracker performs comparatively better. To the best of our knowledge, this is the first time that a quantitative comparison of video object tracking methods is given in terms of their performance on such a large class set. We hope that our benchmark will encourage other researchers to discover ways of mitigating this large performance gap.

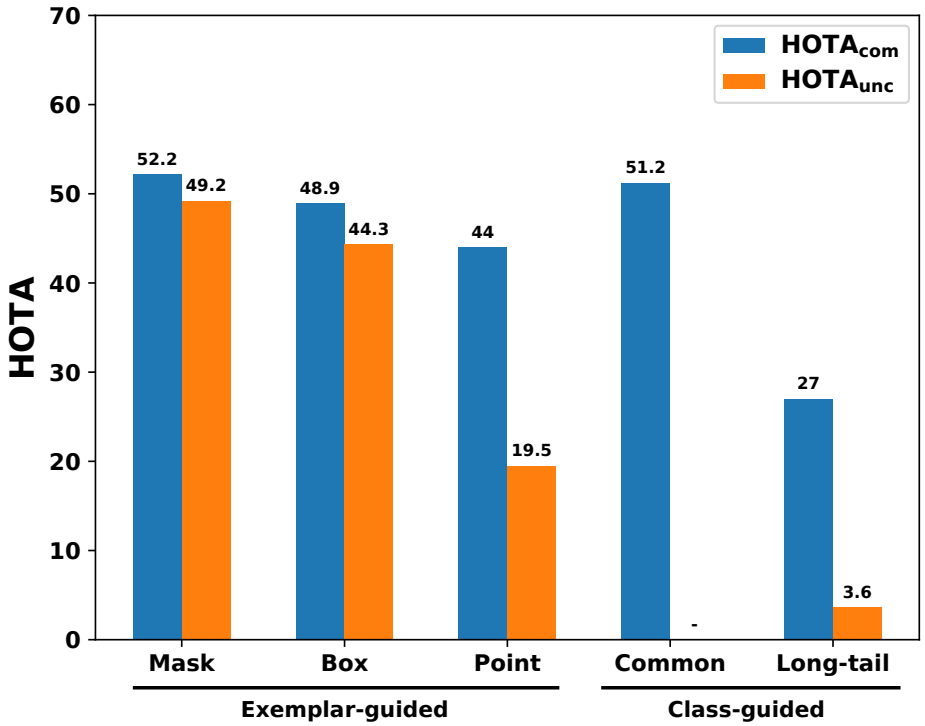


Figure 6.5: Comparison of baseline performances for different tasks.

6.7.2.3 Open-world

Finally, for the open-world task, we use the OWTA metric which is similar to HOTA, but without penalization for false positives. As per the task definition, methods can only be trained on the 'common' class set, but during inference, are expected to additionally segment objects belonging to the 'uncommon' set. Here, we again use the box tracker and STCN tracker with image-level detections from a Mask2Former (Cheng *et al.*, 2021a) model trained on COCO. We additionally report results for the baseline proposed by Liu *et al.* (2022) (OWTB). Unsurprisingly, all methods suffer performance degradation for the 'uncommon' set. The STCN tracker achieves the highest HOTA_{all} score (64.6 on validation), but OWTB performs significantly better in terms of HOTA_{unc} (38.8) compared to the next best baseline (box tracker: 25.0).

6.7.3 Comparison Across Tasks

Using consistent metrics enables us to directly compare results for different methods across different tasks. We illustrate this comparison for our baselines in Fig. 6.5 which charts the best-performing baseline on the validation set for each task. Note that we omitted open-world results since the OWTA metric differs slightly from HOTA. For

HOTA_{com}, the mask exemplar-guided score of 52.2 is only slightly higher than that for the common class-guided task (51.2). This may seem surprising since the exemplar-guided task is inherently easier because, for each target object, methods have access to (1) an explicit cue (mask/box/point), and (2) the class label. However, we noticed that exemplar-guided methods often lose the target object (e.g. due to occlusion or erratic motion), and thereafter cannot recover it; for both STCN and box tracker, if the object is not track-able for more than a certain number of frames, it is assumed to have disappeared. On the other hand, class-guided methods predict an arbitrary number of tracks which may collectively cover more of the ground-truth object. Because HOTA calculation involves Detection Accuracy (DetA), the class-guided methods receive partial credit for correct per-frame detections even if the object ID is inconsistent/fragmented over time. This effect is more visible for BURST because it contains longer videos (∼30s) compared to existing exemplar-guided benchmarks (Pont-Tuset *et al.*, 2017; Xu *et al.*, 2018a) (∼5-10s).

For HOTA_{unc} however, even the worst exemplar-guided score (19.5) is much higher than the 3.6 achieved for class-guided, because exemplar-guided methods are inherently class-agnostic and can propagate arbitrary object masks, but our class-guided tracking-by-detection baselines are very sensitive to per-frame object detection/classification quality, which is currently quite poor for this larger class set.

6.8 Conclusion

We present BURST: a benchmark that unifies six tasks related to object recognition, segmentation, and tracking in video with a clear task taxonomy and consistent evaluation metrics. Our dataset contains a large and diverse video set with pixel-precise masks for a large vocabulary of object classes. We temporally densified the object masks for the training set using a semi-automated pipeline which yields accurate results while drastically reducing human annotation effort. Finally, we presented a number of baselines for the proposed tasks and analyzed their performance. We hope that our benchmark will serve as a valuable resource for researchers to evaluate their object-tracking methods.

The publication on which this chapter is based was presented in 2023. Since then, we have organized a challenge-based workshop at the Conference on Computer Vision and Pattern Recognition (CVPR) called "Tracking and its Many Guises" which was based on the long-tail and open-world tracking tasks in BURST. In terms of adoption by researchers, our first impression based on feedback from the community is that BURST is more interesting for researchers working on existing tasks related to Exemplar-guided tracking such as Video Object Segmentation (VOS) since any VOS method can be readily trained and evaluated on the mask-guided task with no changes. On the other hand, researchers working on class-guided tasks such as Video Instance Segmentation (VIS)

may need to adapt their approaches to train on BURST since our dataset was annotated in a federated manner, i.e. object classes are not exhaustively annotated in every video sequence.

Table 6.2: Dataset Comparison By Size and Difficulty. Comparison of datasets according to various measures of 'difficulty'. Statistics for validation/test may not be exact if data is not publicly available.

Dataset	Setting	Length (hrs)	Masks / Frame	# Classes
VOT (2018)	Single Object	10.7	1	-
DAVIS'17 (2017)	Internet Videos	2.9	2.6	78
YT-VOS (2018a)	Internet Videos	4.5	1.63	94
BDD (2020)	Driving	40	11.4	7
KITTI-MOTS (2019b)	Driving	39.0	5.2	2
MOTS-Chal. (2019b)	Surveillance	34.4	10.0	1
YT-VIS (2019a)	Internet Videos	4.5	1.69	40
UVO (2021a)	Human Actions	3	12.3	-
BURST	General / Diverse	28.9	3.1	482

(a)

Dataset	#Masks	#Tracks	#Frames	#Vids
VOT (2018)	0	0	0	0
DAVIS'17 (2017)	10,238	144	4,219	60
YT-VOS (2018b)	12,918	6,459	94,440	3,471
BDD (2020)	347,442	17,838	30,745	154
KITTI-MOTS (2019b)	38,197	748	8,008	21
MOTS-Chal. (2019b)	26,894	228	2,862	4
YT-VIS (2019a)	103,424	3,774	61,845	2,238
UVO (2021a)	416,001	76,627	39,174	5,641
BURST	318,200	2,645	107,144	500

(b) Statistics for training set

Dataset	#Masks	#Tracks	#Frames	#Vids
VOT (2018)	19,903	62	19,903	62
DAVIS'17 (2017)	16,841	242	6,240	90
YT-VOS (2018b)	4,310	2,155	28,825	1,048
BDD (2020)	77,389	4,873	6,475	32
KITTI-MOTS (2019b)	61,906	961	11,095	28
MOTS-Chal. (2019b)	32,269	328	3044	4
YT-VIS (2019a)	29,431	1,092	17,415	645
UVO (2021a)	177,153	28,271	18,966	5,587
BURST	281,957	13,444	88,569	2,414

(c) Statistics for validation/test sets

Table 6.4: Dataset Comparison by Task. '/' means that the dataset contains annotations to setup the given task, but this is not done officially as part of that benchmark.

Dataset	Exemplar-guided			Class-guided		
	Mask	Box	Point	Common	Long-Tail	Open-World
VOT (2018)	✓	✓	/	✗	✗	✗
DAVIS (2017)	✓	/	/	✗	✗	✗
YouTube-VOS (2018a)	✓	/	/	✗	✗	✗
BDD (2020)	/	/	/	✓	✗	✗
KITTI-MOTS (2019b)	/	/	/	✓	✗	✗
MOTS-Chal. (2019b)	/	/	/	✓	✗	✗
YouTube-VIS (2019a)	/	/	/	✓	✗	✗
OVIS (2022)	/	/	/	✓	✗	✗
UVO (2021a)	/	/	/	✓	✗	✓
BURST	✓	✓	✓	✓	✓	✓

Table 6.5: Baseline results for all tasks using various methods. Evaluation metrics are reported separately for 'common', 'uncommon' and 'all' classes. Object detector training data: *: COCO, †: LVIS.

		Baseline Method	$HOTA_{all}$	$HOTA_{com}$	$HOTA_{unc}$
Exemplar Guided	Mask	STCN (2021c)	49.8	52.2	49.2
		Box Tracker* (2020)	18.0	35.8	13.6
	Box	STCN (PointRend)	45.2	48.9	44.3
		STCN (Matched Det*)	24.5	47.6	18.7
		Box Tracker*	13.7	34.2	8.6
	Point	STCN (Matched Det*)	24.4	44.0	19.5
		Box Tracker*	12.7	31.7	7.9
Class Guided	Common	STCN Tracker*	-	51.2	-
		Box Tracker*	-	45.5	-
	Long-tail	STCN Tracker†	5.5	17.5	2.5
		Box Tracker†	8.2	27.0	3.6
			$OWTA_{all}$	$OWTA_{com}$	$OWTA_{unc}$
	Open-world	STCN Tracker	64.6	71.0	25.0
		Box Tracker	60.9	66.9	24.0
		OWTB (2022)	55.8	59.8	38.8

(a) Results for validation set

		Baseline Method	$HOTA_{all}$	$HOTA_{com}$	$HOTA_{unc}$
Exemplar Guided	Mask	STCN (2021c)	52.4	51.1	52.7
		Box Tracker* (2020)	14.1	28.0	11.4
	Box	STCN (PointRend)	46.0	48.9	45.4
		STCN (Matched Det*)	25.0	41.9	21.7
		Box Tracker*	13.6	27.7	10.8
	Point	STCN (Matched Det*)	24.9	39.5	22.0
		Box Tracker*	10.1	24.4	7.3
Class Guided	Common	STCN Tracker*	-	34.6	-
		Box Tracker*	-	34.3	-
	Long-tail	STCN Tracker†	4.5	17.1	2.0
		Box Tracker†	5.7	20.1	2.9
			$OWTA_{all}$	$OWTA_{com}$	$OWTA_{unc}$
	Open-world	STCN Tracker	57.5	62.9	23.9
		Box Tracker	55.9	61.0	24.6
		OWTB (2022)	56.0	59.9	38.3

(b) Results for test set

7 TarViS: A Unified Approach for Target-based Video Segmentation

7.1 Introduction

In the previous chapter, we proposed a dataset and benchmark to unify multiple tasks related to object segmentation in video under a single umbrella. This chapter can be seen as a logical continuation of that where we propose a network architecture that can tackle multiple video segmentation tasks. To do this, we seek inspiration from several aspects of the works explained in the preceding chapters. For instance, we segment and track objects jointly over space-time by drawing on our experience from STEm-Seg (Chapter 4). Furthermore, we adopt a similar workflow to HODOR (Chapter 5) to encode objects as concise descriptors to tackle the Video Object Segmentation (VOS) task. Finally, we evaluate the Point Exemplar-guided Tracking (PET) task proposed in BURST (Chapter 6) to highlight the generalization capability of our proposed approach.[1]

The ability to understand video scenes has been a long-standing goal of computer vision research because of wide-ranging applications in intelligent vehicles and robots. Early approaches tackled simpler tasks involving contour-based (Kass *et al.*, 1988; Koller *et al.*, 1994) and box-level tracking (Ess *et al.*, 2009; Grabner *et al.*, 2006; Leibe *et al.*, 2008a; Pirsiavash *et al.*, 2011), background subtraction (Elgammal *et al.*, 2000; Stauffer and Grimson, 1999), and motion segmentation (Brox and Malik, 2010; Ochs *et al.*, 2013). The deep learning boom then revolutionized the landscape by enabling methods to perform pixel-precise segmentation on challenging, real-world videos. In the past few years, a number of benchmarks have emerged, which evaluate how well methods can perform video segmentation according to various task formulations. Over time, these

[1]This chapter is based on work published at the Conference on Computer Vision and Pattern Recognition (CVPR) (Athar *et al.*, 2023b). The code implementation and experiments were done by me. All authors contributed useful ideas and engaged in fruitful discussions.

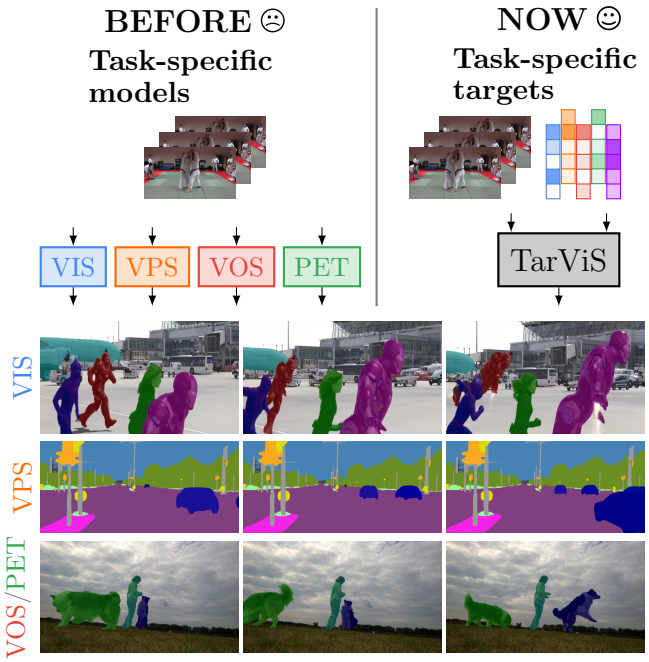


Figure 7.1: Predicted results from a jointly trained TarViS model for four different video segmentation tasks.

tasks/benchmarks have ballooned into separate research sub-communities. Although existing methods are rapidly improving the state-of-the-art for these benchmarks, each of them typically tackles only one narrowly defined task, and generalizing them is non-trivial since the task definition is baked into the core approach.

We argue that this fragmentation is unnecessary because video target segmentation tasks all require the same high-level capability, namely that of identifying, localizing, and tracking rich semantic concepts. Meanwhile, recent progress on Transformer networks has enabled the wider AI research community to move towards unified, multi-task architectures (Alayrac *et al.*, 2022; Jaegle *et al.*, 2021, 2022; Kolesnikov *et al.*, 2022; Reed *et al.*, 2022), because the attention operation (Vaswani *et al.*, 2017) is well-suited for processing feature sets with arbitrary structure and data modality. These developments give us the opportunity to unify the fractured landscape of target-based video segmentation. In this chapter, we propose TarViS: a novel architecture that enables a single, unified model to be *jointly trained for multiple video segmentation tasks*. During inference, the *same model can perform different tasks at runtime* by specifying the segmentation target.

The core idea is that TarViS tackles the generic task of segmenting a set of arbitrary *targets* in video (defined as semantic classes or as specific objects). These targets are encoded as *queries* which, together with the video features, are input to a Transformer-based model. The model iteratively refines these queries and produces a pixel-precise

mask for each target entity. This formulation conceptually fuses all video segmentation tasks (Athar *et al.*, 2023a; Qi *et al.*, 2022; Weber *et al.*, 2021; Yang *et al.*, 2019a) which fall under the umbrella of the above-mentioned generic task, because they differ only in how the *targets* are defined. During both training and inference, TarViS can hot-swap between tasks at run-time by providing the desired target query set.

To demonstrate our generalization capability, we tackle four different tasks: (1) Video Instance Segmentation (VIS) (Qi *et al.*, 2022; Yang *et al.*, 2019a), (2) Video Panoptic Segmentation (VPS) (Kim *et al.*, 2020), (3) Video Object Segmentation (Pont-Tuset *et al.*, 2017), and (4) Point Exemplar-guided Tracking (Athar *et al.*, 2023a) (PET). For VIS, the segmentation targets are all objects in the video belonging to a predefined set of classes. The target set for VPS includes that for VIS, and additionally, a set of non-instantiable *stuff* semantic classes. For VOS, the targets are a specific set of objects for which the first-frame ground-truth mask is provided. PET is a more constrained version of VOS which only provides the location of a single point inside the object, rather than the full object mask.

Existing methods for these tasks lack generalization capability because task-specific assumptions are typically baked into the approach (see Sec. 7.2 and 7.3 for details). In contrast, TarViS can tackle all four tasks with a unified model because we encode the task-specific targets as a set of queries, thus decoupling the network architecture from the task definition. Moreover, our approach can theoretically generalize further, e.g., one could potentially define the target set as all objects described by a given text prompt, though this is beyond the scope of this chapter.

To summarize, our contributions are as follows: we propose TarViS, a novel architecture that can perform any task requiring segmentation of a set of *targets* from video. For the first time, we are able to jointly train and infer a single model on a collection of datasets spanning the four aforementioned tasks (VIS, VPS, VOS, PET). Our experimental results show that TarViS performs competitively for VOS, and achieves state-of-the-art results for VIS, VPS, and PET. Code and pretrained models are available at: https://github.com/Ali2500/TarViS

7.2 Related Work

Multi-task Models. Multi-task learning has a long history (Caruana, 1993) with several architectures and training strategies (Ghiasi *et al.*, 2021; Kirillov *et al.*, 2019; Kokkinos, 2017; Pfeiffer *et al.*, 2019; Ruder, 2017; Zamir *et al.*, 2018). Earlier approaches mostly consist of a shared backbone with fixed task-specific heads, whereas we design a more general architecture for video segmentation with task-specific targets to specify what to segment. Our approach is inspired by recent attention-based models, e.g., PerceiverIO (Jaegle *et al.*, 2021, 2022), which can be trained on diverse data modalities

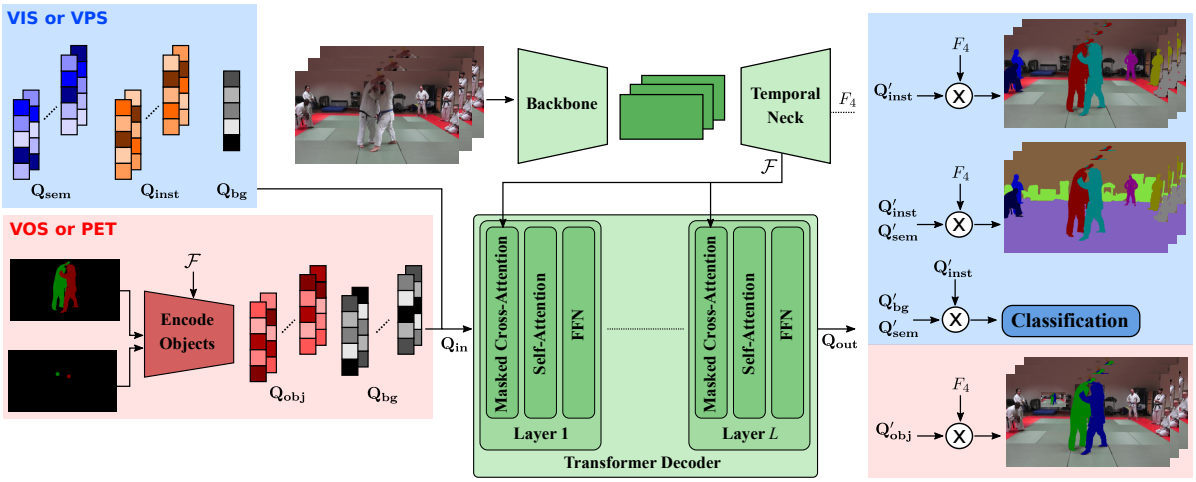


Figure 7.2: TarViS Architecture. Segmentation targets for different tasks are represented by a set of abstract target queries Q_{in}. The core network (in green) is agnostic to the task definitions. The inner product between the output queries Q_{out} and video feature F_4 yields segmentation masks as required by the task.

and task-specific heads are replaced with output queries. UViM (Kolesnikov *et al.*, 2022) follows a similar direction by creating a unified architecture for diverse dense prediction tasks. However, both of these models are trained separately for different tasks. Recent, powerful multi-task vision language models such as Flamingo (Alayrac *et al.*, 2022) and GATO (Reed *et al.*, 2022) tackle a multitude of tasks by requiring a sequence of task-specific input-output examples to prime the model. This is conceptually similar to our task-specific targets, however, our model does not require per-task priming. Moreover, our targets are not modeled as sequence prompts, and we aim for a video segmentation model that is several orders of magnitude smaller. In the realm of video tracking and segmentation, the recently proposed UNICORN (Yan *et al.*, 2022) model tackles multiple object tracking-related tasks with a unified architecture. Unlike TarViS, however, UNICORN follows the task-specific output head approach and is generally oriented towards box-level tracking tasks (Fan *et al.*, 2019; Milan *et al.*, 2016; Muller *et al.*, 2018; Yu *et al.*, 2020), thus requiring non-trivial modifications to tackle VPS or PET.

Query-based Transformer Architectures. Several works (Athar *et al.*, 2022; Carion *et al.*, 2020; Cheng *et al.*, 2021a; Jaegle *et al.*, 2021, 2022; Misra *et al.*, 2021; Wang *et al.*, 2021c; Zhu *et al.*, 2020) use query-based Transformer architectures for various tasks. The workhorse for task learning here is the iterative application of self- and cross-attention, where a set of query vectors (e.g., representing objects) are refined by interacting with each other, and with the input data (e.g., an image). Unlike existing methods which use queries in a task-specific context, TarViS adopts a query-based Transformer architecture in which the queries serve as a mechanism for decoupling the task definition from the

architecture, i.e., our model can learn to tackle different tasks while being agnostic to their definition because the latter is abstracted behind a set of queries.

Task-specific Video Segmentation. Current Video Instance Segmentation (VIS) methods broadly work by predicting object tracks in the video, followed by classification into a pre-defined set of categories. Several approaches (Bertasius and Torresani, 2020; Cao *et al.*, 2020; Fu *et al.*, 2021; Huang *et al.*, 2022; Ke *et al.*, 2021; Lin *et al.*, 2021; Qi *et al.*, 2022; Voigtlaender *et al.*, 2019b; Wu *et al.*, 2022a; Yang *et al.*, 2019a) are based on the tracking-by-detection paradigm, some model video as a joint spatio-temporal volume (Athar *et al.*, 2020, 2021), whereas recent works (Cheng *et al.*, 2021a; Heo *et al.*, 2022; Hwang *et al.*, 2021; Wang *et al.*, 2021c; Wu *et al.*, 2022b) adopt Transformer-based architectures.

For Video Panoptic Segmentation (VPS), methods (Kim *et al.*, 2020; Qiao *et al.*, 2021; Weber *et al.*, 2021) generally extend image-level panoptic approaches (Cheng *et al.*, 2020a) by employing multi-head architectures for semantic segmentation and instance mask regression, classification, and temporal association. In the Video Object Segmentation (VOS) community, state-of-the-art methods are broadly based on the seminal work of Oh *et al.* (2019), which learns *space-time correspondences* between pixels in different video frames, and then uses these to propagate the first-frame masks across the video. Subsequent methods (Cheng and Schwing, 2022; Cheng *et al.*, 2021b,c; Seong *et al.*, 2021; Voigtlaender *et al.*, 2019a; Xie *et al.*, 2021; Yang *et al.*, 2020,?, 2021b) have significantly improved the performance and efficiency of this approach. Point Exemplar-guided Tracking (PET) (Athar *et al.*, 2023a; Homayounfar *et al.*, 2021) is a fairly new task for which the current best approach (Athar *et al.*, 2023a) regresses a pseudo-ground-truth mask from the given point coordinates, and then applies a VOS method (Cheng *et al.*, 2021c) to this mask.

The above methods thus incorporate task-specific assumptions into their core approach. This can benefit per-task performance but makes it difficult for them to generalize across tasks. By contrast, TarViS can tackle all four aforementioned tasks, and generally any target-based video segmentation task, with a single, jointly trained model.

7.3 Method

TarViS can segment arbitrary targets in video since the architecture is flexible with respect to how these targets are defined, thus enabling us to conceptually unify and jointly tackle the four aforementioned tasks (VIS, VPS, VOS, PET). The architecture is illustrated in Fig. 7.2.

For all tasks, the common input to the network is an RGB video clip of length T denoted by $V \in \mathbb{R}^{H \times W \times T \times 3}$. This is input to a 2D backbone network which produces

image-level feature maps, followed by a *Temporal Neck*, which enables feature interaction across time and outputs a set of temporally consistent, multi-scale, D-dimensional feature maps $\mathcal{F} = \{F_{32}, F_{16}, F_8, F_4\}$ where $F_s \in \mathbb{R}^{\frac{H}{s} \times \frac{W}{s} \times T \times D}$. The feature maps $\mathcal{F}$ are then fed to our Transformer decoder, together with a set of queries Q_{in} which represent the segmentation targets. The decoder applies successive layers of self- and masked cross-attention wherein the queries are iteratively refined by attending to each other and to the feature maps, respectively. The refined queries output by the decoder are denoted with Q_{out}. The following subsections explain how TarViS tackles each task in detail.

7.3.1 Video Instance Segmentation

VIS defines the segmentation target set as all objects belonging to a set of predefined classes. Accordingly, the input query set Q_{in} for VIS contains three types of queries: (1) *semantic queries* denoted by $Q_{\text{sem}} \in \mathbb{R}^{C \times D}$ where C is the number of classes defined by the dataset, i.e., each D-dimensional vector in Q_{sem} represents a particular semantic class. (2) *instance queries* denoted by $Q_{\text{inst}} \in \mathbb{R}^{I \times D}$ where I is assumed to be an upper bound on the number of instances in the video clip, and (3) a *background query* denoted by $Q_{\text{bg}} \in \mathbb{R}^{1 \times D}$ to capture inactive instance queries.

The three query sets are concatenated, i.e., $Q_{\text{in}} = \texttt{concat}(Q_{\text{sem}}, Q_{\text{inst}}, Q_{\text{bg}})$, and input to the Transformer decoder, which refines their feature representation through successive attention layers and outputs a set of queries $Q_{\text{out}} = \texttt{concat}(Q'_{\text{sem}}, Q'_{\text{inst}}, Q'_{\text{bg}})$. These are then used to produce temporally consistent instance mask logits by computing the inner product $\langle F_4, Q'_{\text{inst}} \rangle \in \mathbb{R}^{H \times W \times T \times I}$. To obtain classification logits, we compute the inner product $\langle Q'_{\text{inst}}, \texttt{concat}(Q'_{\text{sem}}, Q'_{\text{bg}}) \rangle \in \mathbb{R}^{I \times (C+1)}$.

The three types of queries are initialized randomly at the start of training and optimized thereafter. The instance queries Q_{inst} enable us to segment a varying number of objects from the input clip. During training, we apply Hungarian matching between the predicted and ground-truth instance masks to assign instance queries to video instances, and then supervise their predicted masks and classification logits accordingly. When training on multiple datasets with heterogeneous classes, the semantic query sets are separately initialized per dataset, but Q_{inst} and Q_{bg} are shared.

Comparison to Instance Segmentation Methods. Several Transformer-based methods (Carion *et al.*, 2020; Cheng *et al.*, 2021a; Wang *et al.*, 2021c; Zhu *et al.*, 2020) for image/video instance segmentation also use queries to segment a variable number of input instances. The key difference to our approach is the handling of object classes: existing works employ only instance queries which are input to a fully-connected layer with a fan-out of $C + 1$ to obtain classification (and background) scores. The notion of class-guided instance segmentation is thus baked into the approach. By contrast, TarViS is agnostic to the task-specific notion of object classes because it models them as

arbitrary queries that are dynamic inputs to the network. The semantic representation for these queries is thus decoupled from the core architecture and is only learned via loss supervision. An important enabler for this approach is the background query Q_{bg}, which serves as a 'catch-all' class to represent everything that is not in Q_{sem}. It is used to classify non-active instance queries, and its mask logits are supervised to segment all non-object input pixels.

7.3.2 Video Panoptic Segmentation

VPS defines the segmentation targets as all objects belonging to a set of *thing* classes (e.g., 'person', 'car'), and additionally, a set of non-instantiable *stuff* classes (e.g., 'sky', 'grass') which cover all non-object pixels. TarViS can tackle VPS with virtually no modification to the workflow in Sec. 7.3.1. We can compute semantic segmentation masks for the input clip by simply taking the inner product between Q_{sem} and the video features: $\langle F_4, Q'_{\text{sem}} \rangle \in \mathbb{R}^{H \times W \times T \times C}$. Note that here, Q_{sem} contains queries representing both *thing* and *stuff* classes.

Comparison to VPS Methods. Current VPS datasets (Kim *et al.*, 2020; Weber *et al.*, 2021) involve driving scene videos captured from moving vehicles. Methods tackling this task (Kim *et al.*, 2020; Qiao *et al.*, 2021) are based on earlier image panoptic segmentation approaches (Cheng *et al.*, 2020a) which involve multi-head networks for semantic and instance segmentation prediction. In terms of image-level panoptic segmentation, Mask2Former (Cheng *et al.*, 2021a) uses a Transformer-based architecture, but it models *stuff* classes as instances that are Hungarian-matched to the ground-truth target during training, whereas TarViS models semantic classes and instances using separate, designated queries.

7.3.3 Video Object Segmentation and Point Exemplar-guided Tracking

VOS and PET are instantiations of a general task where the segmentation targets are a set of O objects for which some ground-truth cue $\mathcal{G}$ is given. For VOS, $\mathcal{G}$ is provided as the first-frame object masks $M_{\text{obj}} \in \mathbb{R}^{O \times H \times W}$, whereas for PET, $\mathcal{G}$ is provided as the (x, y) coordinates $P_{\text{obj}} \in \mathbb{R}^{O \times 2}$ of a point inside each of the objects. TarViS jointly tackles both tasks by adopting a generalized approach in which the O target objects are encoded into a set of *object queries* Q_{obj}. Thus, both VOS and PET boil down to designing a function $\texttt{EncodeObjects}(\cdot)$ which regresses Q_{obj} from the ground-truth cues $\mathcal{G}$ and feature maps $\mathcal{F}$:

$$Q_{\text{obj}} \leftarrow \texttt{EncodeObjects}(\mathcal{G}, \mathcal{F}). \tag{7.1}$$

Note that Q_{obj} is conceptually analogous to Q_{sem} and Q_{inst} used for VIS in that all three are abstract representations for their respective task-specific segmentation targets.

Video Object Segmentation. We seek inspiration from HODOR (Athar *et al.*, 2022) to implement `EncodeObjects` for VOS, a recent method for weakly-supervised VOS, which encodes objects into concise *descriptors* as follows: the descriptors are initialized by average pooling the image features inside the object masks, followed by iterative refinement where the descriptors attend to each other (self-attention) and to their respective soft-masked image features (cross-attention).

For TarViS, we employ a lightweight *Object Encoder* with a similar workflow to encode the objects as a set of queries Q_{obj}, but with two differences to HODOR (Athar *et al.*, 2022): instead of cross-attending to the entire image feature map ($H \cdot W$ points) with soft-masked attention, we apply hard-masked cross-attention to at most $p_{\max}$ feature points per object, where $p_{\max} \ll H \cdot W$. Object masks containing more than $p_{\max}$ points are sub-sampled accordingly. This significantly improves the memory/run-time overhead of our Object Encoder. Secondly, we note that the process of distilling object features into a single descriptor involves a loss of object appearance information, which degrades performance. We, therefore, model each object with q_o queries (instead of one) by spatially dividing each object mask into q_o segments, i.e., $Q_{\text{obj}} \in \mathbb{R}^{O \times q_o \times D}$ (we use $q_o = 4$).

In addition to Q_{obj}, we initialize a set of background queries $Q_{\text{bg}} \in \mathbb{R}^{B \times D}$ to model the non-target pixels in the reference frame. Following HODOR (Athar *et al.*, 2022), we employ multiple background queries, which are initialized dynamically by dividing the video frame containing the ground-truth masks M_{obj} into a 4×4 grid and average pooling the non-object pixels in each grid cell. The Object Encoder jointly refines the background and object queries to yield $Q_{\text{in}} = \texttt{concat}(Q_{\text{obj}}, Q_{\text{bg}})$. During training, the mask logits for the multiple background queries are aggregated per pixel by applying $\texttt{max}(\cdot)$ and supervised to segment all pixels not part of the target object set.

The remaining workflow follows that for VIS and VPS: Q_{in} is input to the Transformer decoder together with the video features $\mathcal{F}$. The refined output query set $Q_{\text{out}} = \texttt{concat}(Q'_{\text{obj}}, Q'_{\text{bg}})$ is then used to compute the inner product $\langle F_4, Q'_{\text{obj}} \rangle \in \mathbb{R}^{H \times W \times T \times O \times q_o}$. Subsequently, $\texttt{max}(\cdot)$ is applied on the q_o-sized dimension to obtain the final mask logits for the O target objects.

Point Exemplar-guided Tracking. For PET we implement `EncodeObjects` in the exactly same way as VOS: the given point coordinates P_{obj} are converted into a mask with just one non-zero pixel, followed by iterative refinement by the Object Encoder (with shared weights for VOS and PET). The only difference is that here we represent each of the O objects with just one query, i.e., $Q_{\text{obj}} \in \mathbb{R}^{O \times D}$ ($q_o = 1$). The subsequent workflow is also identical to that for VOS: the queries are refined by the Transformer decoder followed by an inner product with F_4 to obtain object mask logits.

Comparison to VOS and PET Methods. Current state-of-the-art VOS methods are largely based on STM (Oh *et al.*, 2019). It involves learning pixel-to-pixel correspondences across video frames, which are then used to propagate the given object mask across the video. This approach is effective since every pixel in the given mask can be individually mapped to future frames, thus preserving fine-grained object details. The core approach is, however, task-specific since it assumes the availability of first-frame object masks, and does not generalize to the PET (see Sec. 7.4.2). PET can be viewed as a more constrained version of VOS, where only a single object point is provided instead of the full mask. Consequently, PET (Athar *et al.*, 2023a) is currently tackled by casting it as a VOS problem by using an image instance segmentation network (Cheng *et al.*, 2021a) to regress pseudo-ground-truth object masks from the given point coordinates P_{obj}.

On the other hand, our approach of encoding objects as concise queries causes loss of fine-grained object appearance information, but it has the advantage of being agnostic to how $\mathcal{G}$ is defined. As evident from the unified workflow for VOS and PET, any variation of these tasks with arbitrary ground-truth cues $\mathcal{G}$ can be seamlessly fused into our architecture as long as we can implement an effective `EncodeObjects` function to regress Q_{obj} from the given $\mathcal{G}$.

7.3.4 Network Architecture

Table 7.1: Results for Video Instance Segmentation (VIS) on the YouTube-VIS 2021 (Yang *et al.*, 2019a) and OVIS (Qi *et al.*, 2022) validation sets. Mask2Former is abbreviated as 'M2F'.

Method	Backbone	Shared Model	YouTube-VIS 2021					OVIS				
			AP	AP50	AP75	AR1	AR10	AP	AP50	AP75	AR1	AR10
M2F-VIS (2021a)	R-50	✗	40.6	60.9	41.8	-	-	-	-	-	-	-
IDOL (2022a)	R-50	✗	43.9	68.0	49.6	38.0	50.9	30.2	51.3	30.0	15.0	37.5
MinVIS (2022)	R-50	✗	44.2	66.0	48.1	39.2	51.7	25.0	45.5	24.0	13.9	29.7
VITA (2022)	R-50	✗	45.7	67.4	49.5	**40.9**	53.6	19.6	41.2	17.4	11.7	26.0
TarViS	R-50	✓	**48.3**	**69.6**	**53.2**	40.5	**55.9**	**31.1**	**52.5**	**30.4**	**15.9**	**39.9**
M2F-VIS (2021a)	Swin-T	✗	45.9	68.7	50.7	-	-	-	-	-	-	-
TarViS	Swin-T	✓	**50.9**	**71.6**	**56.6**	42.2	57.2	34.0	55.0	34.4	16.1	40.9
IDOL (2022a)	Swin-L	✗	56.1	80.8	63.5	45.0	60.1	42.6	65.7	**45.2**	17.9	49.6
VITA (2022)	Swin-L	✗	57.5	80.6	61.0	47.7	62.6	27.7	51.9	24.9	14.9	33.0
TarViS	Swin-L	✓	**60.2**	**81.4**	**67.6**	**47.6**	**64.8**	**43.2**	**67.8**	44.6	**18.0**	**50.4**

Temporal Neck. TarViS produces target masks by computing the inner product between Q_{out} and the video feature map F_4. For this to work, the per-pixel features $\mathcal{F}$ must be aligned for the same, and dissimilar for different targets. Some image instance segmentation methods (Cheng *et al.*, 2021a; Zhu *et al.*, 2020) apply *Deformable Attention* (Zhu *et al.*, 2020) to the backbone feature maps to efficiently learn multi-scale image

Table 7.2: Video Panoptic Segmentation (VPS) results for validation sets of KITTI-STEP (Weber *et al.*, 2021), CityscapesVPS (Kim *et al.*, 2020) and VIPSeg (Miao *et al.*, 2022).

Method	Shared Model	KITTI-STEP			CityscapesVPS			VIPSeg			
		STQ	AQ	SQ	VPQ	VPQ^{Th}	VPQ^{St}	VPQ	VPQ^{Th}	VPQ^{St}	STQ
Mask Propagation (2021)	✗	0.67	0.63	0.71	-	-	-	-	-	-	
Track (2020)	✗	-	-	-	55.9	43.7	64.8	-	-	-	
VPSNet (2020)	✗	0.56	0.52	0.61	57.0	44.7	66.0	14.0	14.0	14.2	20.8
VPSNet-SiamTrack (2021)	✗	-	-	-	57.3	44.7	66.4	17.2	17.3	17.3	21.1
VIP-Deeplab (2021)	✗	-	-	-	**63.1**	**49.5**	**73.0**	16.0	12.3	18.2	22.0
Clip-PanoFCN (2022)	✗	-	-	-	-	-	-	22.9	25.0	20.8	31.5
TarViS (R-50)	✓	0.70	0.70	0.69	53.3	35.9	66.0	33.5	39.2	28.5	43.1
TarViS (Swin-T)	✓	0.71	0.71	0.70	58.0	42.9	69.0	35.8	42.7	29.7	45.3
TarViS (Swin-L)	✓	**0.72**	**0.72**	**0.73**	58.9	43.7	69.9	**48.0**	**58.2**	**39.0**	**52.9**

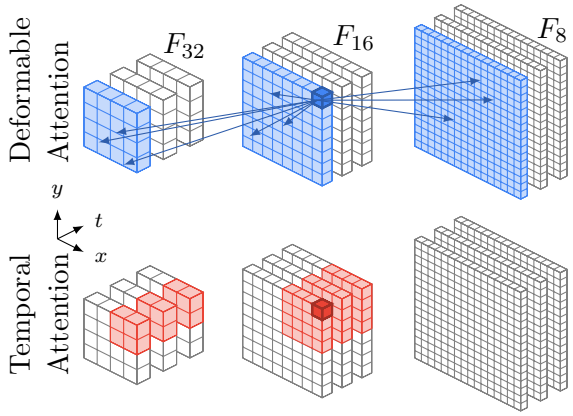


Figure 7.3: Temporal Neck Layer. Colored regions denote the attention field w.r.t the selected pixel (darkened). Deformable Attention is spatially unrestricted but temporally limited to a single frame, whereas Temporal Attention is spatially localized, but temporally unrestricted. F_8 is inactive for the temporal attention.

features. For TarViS, however, the features must also be temporally consistent across the entire input video clip. To achieve this, we propose a novel *Temporal Neck* architecture inspired by the work of *et al.* Bertasius *et al.* (2021) for video action classification. We enable efficient spatio-temporal feature interaction by applying two types of self-attention in an alternating fashion: the first is spatially global and temporally localized, whereas the second is spatially localized and temporally global. The first operation is implemented with Deformable Attention, following existing work (Cheng *et al.*, 2021a; Zhu *et al.*, 2020). The second operation, Temporal Attention, involves dividing the input space-time volume into a grid along the spatial axes and then applying self-attention to the space-time feature volume inside each grid cell. Both operations allow feature interaction across multiple scales. Both attention operations are illustrated in Fig. 7.3. We exclude F_8 from temporal attention since we found this to be more memory-efficient without negatively impacting prediction quality.

Transformer Decoder. The decoder architecture follows that of Mask2Former (Cheng *et al.*, 2021a): the input queries are iteratively refined over multiple layers. In each layer, the queries first cross-attend to their respective masked video features, then self-attend to each other, followed by feed-forward layers.

7.3.5 Inference

To infer on videos with arbitrary length, we split videos into clips of length T_{clip} with an overlap of T_{ov} between successive clips. Object tracks are associated across clips based on their mask IoU in the overlapping frames. For the VOS tasks, the object queries for an intermediate clip are initialized by using the predicted masks in the overlapping frames from the previous clip as a pseudo-ground-truth. For VPS, we average the semantic segmentation logits in the overlapping frames. Our approach is thus *near-online* because the time delay in obtaining the output for a given frame is at most $T_{\text{clip}} - T_{\text{ov}} - 1$ (except for the first clip in the video).

7.4 Experiments

7.4.1 Implementation Details

Our Temporal Neck contains 6 layers of Deformable and Temporal Attention. We pre-train for 500k iterations on pseudo-video clips generated by applying on-the-fly augmentations to images from COCO (Lin *et al.*, 2014), ADE20k (Zhou *et al.*, 2017), Mapillary (Neuhold *et al.*, 2017) and Cityscapes (Cordts *et al.*, 2016). The samples are either trained for VPS, VIS, VOS, or PET. This is followed by fine-tuning for 90k iterations jointly on samples from YouTube-VIS (Yang *et al.*, 2019a), OVIS (Qi *et al.*, 2022), KITTI-STEP (Weber *et al.*, 2021), CityscapesVPS (Kim *et al.*, 2020), VIPSeg (Miao *et al.*, 2022), DAVIS (Pont-Tuset *et al.*, 2017) and BURST (Athar *et al.*, 2023a). We train on 32 Nvidia A100 GPUs with batch size 1 per GPU. For each of the query types (Q_{sem}, Q_{inst}, Q_{obj}, Q_{bg}) discussed in Sec. 7.3, we employ a learned query embedding, which is used when computing the $\texttt{Key}^T\texttt{Query}$ affinity matrix for multi-head attention inside the decoder. We refer to the supplementary for more details.

7.4.2 Benchmark Results

All results are computed with a single, jointly trained model that performs different tasks by simply providing the corresponding query set at run-time.

Video Instance Segmentation (VIS). We evaluate on (1) YouTube-VIS 2021 (Yang *et al.*, 2019a) which covers 40 object classes and contains 2985/421 videos for training/-validation, and (2) OVIS (Qi *et al.*, 2022) which covers 25 object classes. It contains 607/140 videos for training/validation which are comparatively longer and more occluded. The AP scores for both are reported in Tab. 7.1. For all three backbones, TarViS achieves state-of-the-art results for both benchmarks even though other methods are trained separately per benchmark whereas we use a single model. On YouTube-VIS, TarViS achieves 48.3 AP with a ResNet-50 backbone compared to the 45.7 achieved by VITA (Heo *et al.*, 2022). With Swin-L, we achieve 60.2 AP which is also higher than the 57.5 by VITA. On OVIS with ResNet-50, our 31.1 AP is higher than the 30.2 for IDOL (Wu *et al.*, 2022a), and with Swin-L, TarViS (43.2 AP) outperforms the current state-of-the-art IDOL (42.6 AP).

Video Panoptic Segmentation (VPS). We evaluate VPS on three datasets: (1) KITTI-STEP (Weber *et al.*, 2021), which contains 12/9 lengthy driving scene videos for training/validation with 19 semantic classes (2 *thing* and 17 *stuff* classes), (2) CityscapesVPS (Kim *et al.*, 2020), which contains 50 short driving scene clips, each with 6 annotated frames, and (3) VIPSeg (Miao *et al.*, 2022), which is a larger dataset with 2806/343 in-the-wild videos for training/validation and 124 semantic classes. The results are reported in Tab. 7.2. For KITTI-STEP, TarViS achieves 70% STQ with a ResNet-50 backbone which is better than all existing approaches. The performance further improves to 72% with Swin-L. For CityscapesVPS, TarViS achieves 58.9 VPQ which is higher than all other methods except VIP-Deeplab (Qiao *et al.*, 2021) (63.1).

For VIPSeg, TarViS outperforms existing approaches by a significant margin. With a ResNet-50 backbone, our 33.5 VPQ is 10.6% higher than the 22.9 by Clip-PanoFCN (Miao *et al.*, 2022). With a Swin-Large backbone, TarViS achieves 48.0 VPQ which is more than double that of Clip-PanoFCN (22.9). Note that VIP-Deeplab performs significantly worse for VIPSeg (16.0 VPQ), showing that TarViS generalizes better across benchmarks. Finally, we note that larger backbones result in significant performance gains for datasets with in-the-wild internet videos as in VIPSeg, but for specialized driving scene datasets (e.g. KITTI-STEP and Cityscapes-VPS), the improvements are much smaller.

Video Object Segmentation (VOS). We evaluate VOS on the DAVIS 2017 (Pont-Tuset *et al.*, 2017) dataset, which contains 60/30 YouTube videos for training/validation. The results in Tab. 7.3 show that TarViS achieves 85.3 $\mathcal{J}\&\mathcal{F}$ which is higher than all existing methods except STCN (Cheng *et al.*, 2021c) (85.4) and XMem (Cheng and Schwing, 2022) (86.2). As mentioned in Sec. 7.3.3, the process of encoding objects as queries incurs a loss of fine-grained information, which is detrimental to performance. On the other hand, space-time correspondence (STC) based approaches learn pixel-to-pixel affinities between frames, which enables them to propagate fine-grained object appearance information. We note, however, that TarViS is the first method not based on

Table 7.3: Results for VOS on DAVIS (Pont-Tuset *et al.*, 2017) and PET on BURST (Athar *et al.*, 2023a). Detailed PET metrics are provided in supplementary.

Method	DAVIS (VOS)			BURST (PET)	
	$\mathcal{J}\&\mathcal{F}$	$\mathcal{J}$	$\mathcal{F}$	$\text{H}_{\text{all}}^{\text{val}}$	$\text{H}_{\text{all}}^{\text{test}}$
UNICORN∗ (2022)	70.6	66.1	75.0	-	-
HODOR (2022)	81.3	78.4	83.9	-	-
STM (2019)	81.8	79.2	84.3	-	-
CFBI (2020)	81.9	79.1	84.6	-	-
HMMN (2021)	84.7	81.9	87.5	-	-
AOT (2021b)	84.9	82.3	87.5	-	-
STCN (2021c)	85.4	82.2	88.6	-	-
XMem (2022)	**86.2**	**82.9**	**89.5**	-	-
Box Tracker (2020)	-	-	-	12.7	10.1
STCN+M2F (2021a; 2021c)	-	-	-	24.4	24.9
TarViS (R-50)	82.6	79.3	85.9	30.9	32.1
TarViS (Swin-T)	82.8	79.6	86.0	36.0	**36.4**
TarViS (Swin-L)	85.3	81.7	88.5	**37.5**	36.1

the STC paradigm which achieves this level is performance (85.3 $\mathcal{J}\&\mathcal{F}$), outperforming several STC-based methods as well as all non-STC based methods e.g. HODOR (Athar *et al.*, 2022) (81.5) and UNICORN (Yan *et al.*, 2022) (70.6).

Point Exemplar-guided Tracking (PET). PET is evaluated on the recently introduced BURST benchmark (Athar *et al.*, 2023a) which contains 500/1000/1500 diverse videos for training/validation/testing. It is a constrained version of VOS which only provides the point coordinates of the object mask centroid instead of the full mask. Tab. 7.3 shows that existing methods can only tackle either VOS or PET. To verify this, we tried adapting STCN (Cheng *et al.*, 2021c) for PET by training it with point masks, but the training did not converge. By contrast, TarViS encodes objects into queries, which enables it to tackle both tasks with a single model since the object guidance (point or mask) is abstracted behind the `EncodeObjects`$(\cdot)$ function.

TarViS achieves a HOTA_{all} score of 37.5 and 36.4 on the validation and test sets, respectively, which is significantly better than the 24.4 and 24.9 achieved by the best-performing baseline method which casts PET as a VOS problem by regressing a pseudo-ground-truth mask from the given point, followed by applying a VOS approach (STCN (Cheng *et al.*, 2021c)).

7.4.3 Ablations

Table 7.4 shows several architecture/training ablations.

Table 7.4: Ablation experiment results with ResNet-50 backbone. C-VPS: CityscapesVPS, YTVIS: YouTube-VIS, KITTI: KITTI-STEP.

	Setting	Video Training Data							VIS		VPS			VOS	PET
		YTVIS	OVIS	KITTI	C-VPS	VIPSeg	DAVIS	BURST	YTVIS (mAP)	OVIS (mAP)	KITTI (STQ)	C-VPS (VPQ)	VIPSeg (VPQ)	DAVIS ($\mathcal{J}\&\mathcal{F}$)	BURST ($\text{HOTA}^{\text{val}}_{\text{all}}$)
1.	VIS	✓	✓						46.3	31.5	-	-	-	-	-
2.	VPS			✓	✓	✓			-	-	0.70	49.7	32.4	-	-
3.	VOS + PET						✓	✓	-	-	-	-	-	81.1	34.7
4.	No Semantic Queries	✓	✓						44.7	29.8	-	-	-	-	-
5.	No Temporal Neck	✓	✓	✓	✓	✓	✓	✓	42.8	22.3	0.69	51.2	28.9	78.7	30.3
6.	Final	✓	✓	✓	✓	✓	✓	✓	48.3	31.1	0.70	53.3	33.5	82.0	30.9

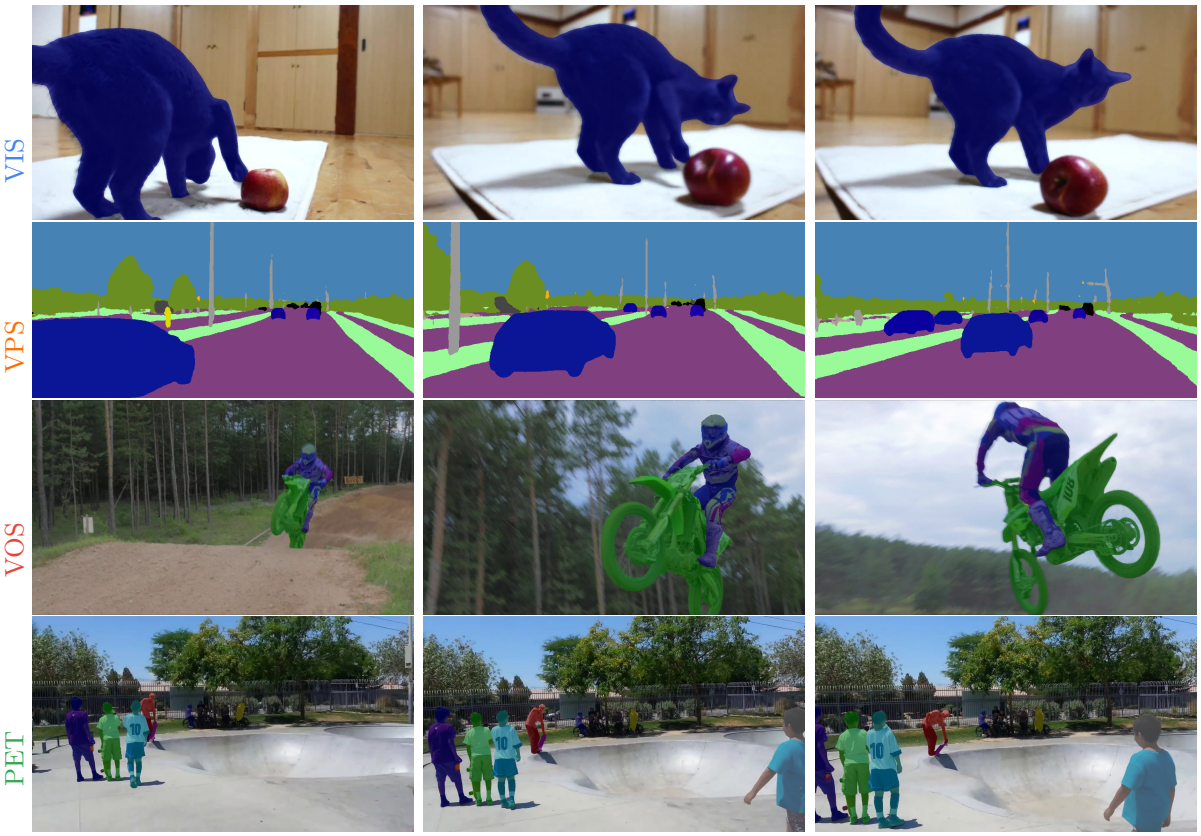


Figure 7.4: Qualitative results from a single TarViS model for all four tasks. Further results are shown in the supplementary.

Task-specific Training (row 1-3). The first three rows show results for task-specific models. We train a single model for VOS and PET since both tasks are closely related. We note that the VIS-only model performs worse than the multi-task model on YouTube-VIS (46.3 vs. 48.3) but slightly better on OVIS (31.5 vs. 31.1). For VPS,

the performance on KITTI-STEP is unchanged, but Cityscapes-VPS and VIPSeg both show improvements with the multi-task model. Lastly, for VOS the task-specific model performs slightly worse on DAVIS (81.1 vs. 82.0) but significantly better on BURST for PET (34.7 vs. 30.9). To summarize, the final, multi-task model performs better on 4/7 benchmarks, worse on 2/7, and matches performance on 1/7 when compared to task-specific models. We thus conclude that the combination of multi-task supervision and more data is generally beneficial for performance.

Semantic Queries for VIS (row 4). TarViS represents object classes as dynamic query inputs to the network (Q_{sem}, Sec. 7.3.1). We ablate this by modifying our network to work for only VIS by discarding the semantic/background queries and adopting a technique similar to existing methods (Cheng *et al.*, 2021a; Wang *et al.*, 2021c), i.e. using instance queries Q_{inst} in conjunction with a linear layer for classification (separate for each dataset). Comparing the results with the VIS-only setting which is trained on similar data, we see that this architecture performs worse than the VIS-only setting on both YouTube-VIS (44.7 vs. 46.3) and OVIS (29.8 vs. 31.5). Thus, our semantic query based classification makes the network architecture task-agnostic and also yields better performance.

Temporal Neck (row 5). We validate our novel Temporal Neck (Sec. 7.3.4) by training a model with a simpler neck that contains only Deformable Attention layers (Zhu *et al.*, 2020), similar to Mask2Former (Cheng *et al.*, 2021a), i.e. there is no feature interaction across frames. Doing this degrades performance across all datasets, with particularly large drops for YouTube-VIS (42.8 vs. 48.3) and OVIS (22.3 vs. 31.1). This shows that inter-frame feature interactions enabled by our Temporal Neck are highly beneficial for downstream tasks.

7.5 Discussion

Limitations. Training on multiple datasets/tasks does not necessarily improve performance on all benchmarks. For VOS, the model exhibits class bias and sometimes fails to track unusual objects that were not seen during training.

Future Outlook. We jointly trained TarViS for four different tasks to validate its generalization capability. The architecture can, however, tackle any video segmentation task for which the targets can be encoded as queries. The recent emergence of joint language-vision models (Li *et al.*, 2022; Radford *et al.*, 2021; Ramesh *et al.*, 2021) thus makes it possible to perform multi-object segmentation based on a text prompt if the latter can be encoded as a target query using a language encoder (Devlin *et al.*, 2019). Another interesting possibility is that TarViS could be applied to multiple tasks *in the same forward pass* by simply concatenating the task-specific queries. Fig. 7.5



Figure 7.5: TarViS performing VIS and VOS in a single forward pass. We provide the mask for the dragon on the left, and the semantic query for the 'person' class.

offers a promising outlook for this; it shows our model's output for a video clip from a popular TV series where we perform VIS and VOS simultaneously by providing the semantic query for the 'person' class (from YouTube-VIS (Yang *et al.*, 2019a)), and the VOS-based object queries for the dragon by annotating its first frame mask, i.e. $Q_{\text{in}} = \texttt{concat}(Q_{\text{sem}}, Q_{\text{inst}}, Q_{\text{obj}}, Q_{\text{bg}})$. TarViS successfully segments all four persons in the scene (VIS) and the dragon (VOS), even though our model was never trained to simultaneously tackle both tasks in a single forward pass.

7.6 Query Visualization

To gain some insight into the feature representation learned by TarViS for different targets, we provide visualizations of the target queries for various tasks and input video clips in Fig. 7.6,7.7,7.8. The setup is as follows: for each video clip, we run inference twice: (1) as VIS where the targets are all instances belonging to the 40 object classes from YouTube-VIS (Yang *et al.*, 2019a), and (2) as VOS by providing the first-frame mask for the objects. We deliberately used videos where the set of ground-truth objects would be the same for both tasks. The plot on the right visualizes the union of the target query set for both runs by projecting them from 256 dimensions down to 2 using PCA. The image tile on the left shows our model's predicted masks for the target objects (the

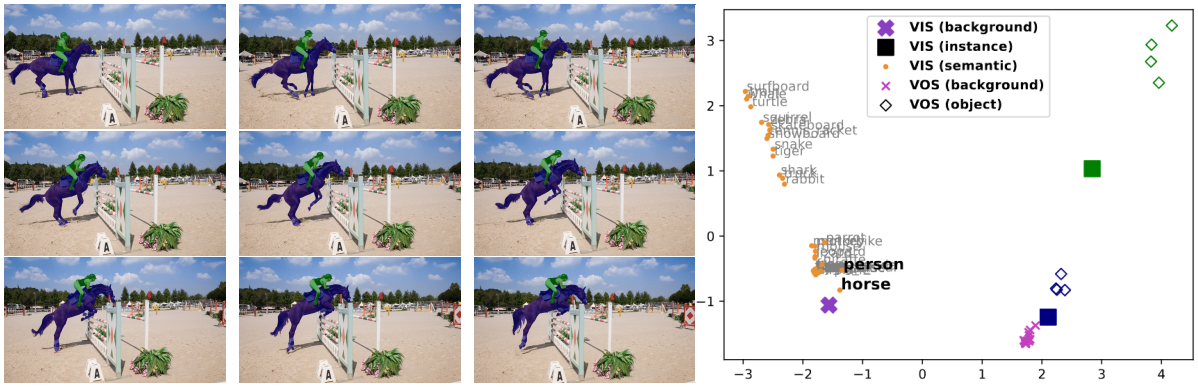


Figure 7.6: Target query visualization for the 'horsejump-high' sequence in DAVIS.

prediction quality for these videos is very good for both VIS and VOS, so we choose one set of results arbitrarily).

For ease of understanding, we use fixed colors for semantic and background queries (as indicated in the plot legend). For the object queries (VOS) and instance queries (VIS), the color of the query point is consistent with the color of the object mask in the image tile. Note that for VOS we used $q_o = 4$ object queries per target, hence there are 4 hollow diamond-shaped points per object.

We stress that not all aspects of these plots are intuitively explainable. The main limitation here is the harsh dimensionality reduction from 256 dimensions to 2. Some speculative intuition based on the plots is as follows:

- The internal representation for a given object is generally consistent across tasks. As an example, consider the horse and person targets in Fig. 7.6: we note that the green query points (person) are close to each other for both VIS and VOS. Likewise, the blue query points (horse) follow the same behavior.
- The network devotes a large portion of the feature space for instances/objects, and relatively less for the various semantic classes. As seen in all three plots, the semantic queries are tightly clustered together, whereas the instance/object queries are spread out over a larger span of the feature space.

Iterative Evolution of Feature Representation. Fig. 7.9 shows a side-by-side visualization of how the query feature representation evolves inside the transformer decoder as it iteratively refined the queries using multiple attention layers. The plot on the left shows the queries at the 'zeroth' layer (i.e. prior to any interaction with the video features), and the plot on the right shows the final output queries from the last layer (these are identical to the plot in Fig. 7.6 except for the axes range). We note that the distance between the queries for the two objects increases after refinement and that the semantic queries are also slightly more spaced out after refinement.

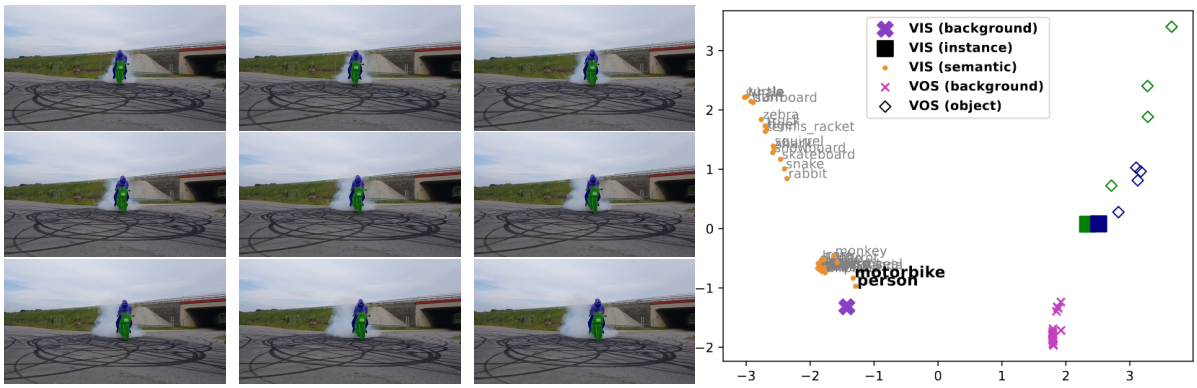


Figure 7.7: Target query visualization for the 'mbike-trick' sequence in DAVIS.

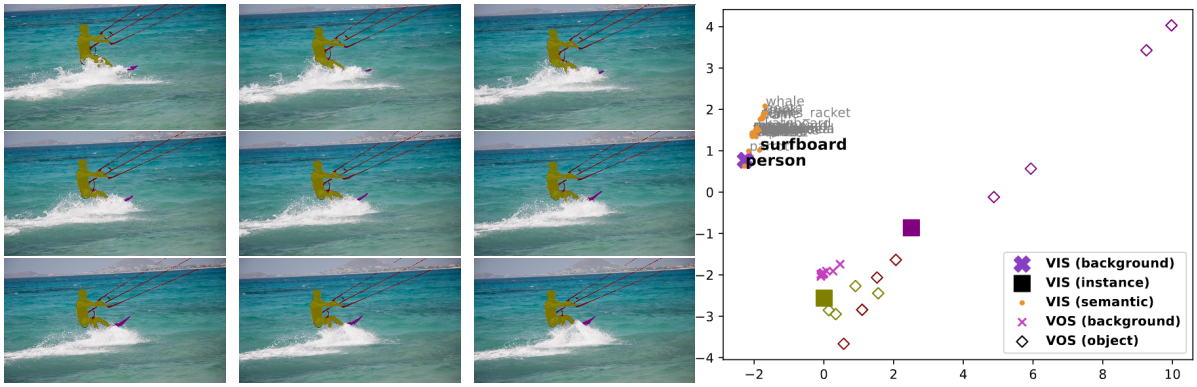
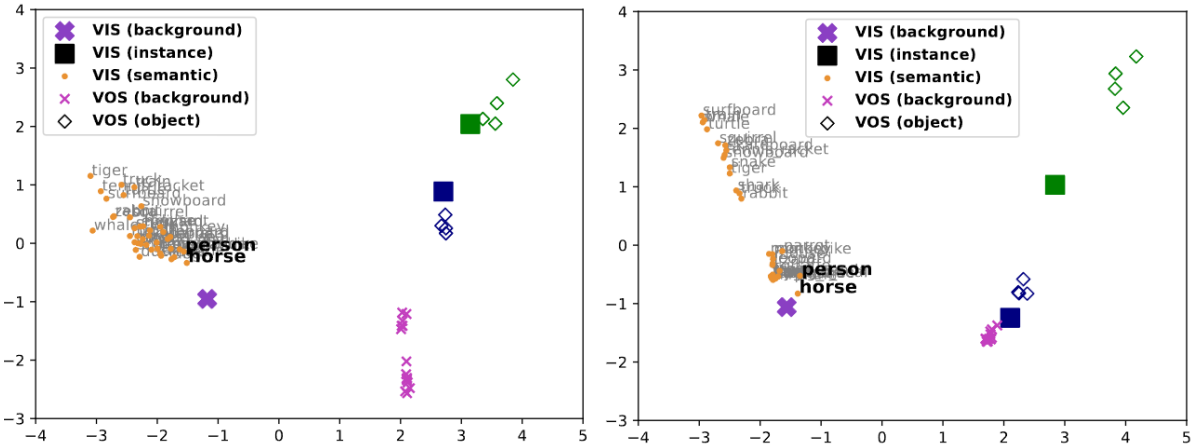


Figure 7.8: Target query visualization for the 'kitesurf' sequence in DAVIS.



(a) First layer queries. **(b)** Last layer queries.

Figure 7.9: Evolution of the different queries from the first layer to the last layer of the transformer decoder. Queries correspond to the 'horsejump-high' video from DAVIS as shown in Figure 7.6

7.7 Conclusion

We presented TarViS: a novel, unified approach for tackling any task requiring pixel-precise segmentation of a set of *targets* in video. We adopt a generalized paradigm where the task-specific targets are encoded into a set of *queries* which are then input to our network together with the video features. The network is trained to produce segmentation masks for each target entity but is inherently agnostic to the task-specific definition of these targets. To demonstrate the effectiveness of our approach, we applied it to four different video segmentation tasks (VIS, VPS, VOS, PET). We showed that a single TarViS model can be jointly trained for all tasks, and during inference can hot-swap between tasks without any task-specific fine-tuning. Our model achieved state-of-the-art performance on five benchmarks and has multiple, promising directions for future work.

In the preceding chapters, we tackled multiple conceptually overlapping tasks and also proposed a dataset/benchmark to bring these tasks together. The work in this chapter can therefore be seen as a culmination of previous works where we present a single approach multi-task capability that draws on ideas and network components from STEm-Seg (Chapter 4) and HODOR (Chapter 5) and is evaluated on tasks from BURST (Chapter 6). The publication on which this chapter is based was given a unanimous 'strong accept' rating from all reviewers and was well-received by the community. It was accepted as a 'Highlight' paper which was a status granted to only the top 2.6% of submissions.

8 Conclusion

The objective of this thesis was to have an impactful contribution to research pertaining to object tracking and segmentation in monocular camera video. To achieve this, we proposed multiple network architectures and datasets which tackled specific tasks from a novel angle, and generally also improved the state-of-the-art on them. In the following paragraphs, we briefly recap the contributions in Chapters 4-7.

Chapter 4: Here we proposed STEm-Seg, which is a novel end-to-end trainable approach for instance segmentation and tracking in video. To achieve this, we learned spatio-temporal embeddings for each pixel in the given input video which are optimized to be similar for pixels belonging to the same object, and dissimilar for pixels belonging to different objects/backgrounds. Specifically, we model the pixel embeddings belonging to an object using a Gaussian distribution and predict the variance and instance center likelihood at each pixel, in addition to the embeddings. This formulation enabled efficient and end-to-end learned inference since the model predicts all the parameters required to cluster the embeddings, thereby avoiding the need for slow and non-learned clustering algorithms. Moreover, our interpretation of embeddings as offsets to the spatio-temporal coordinates of the given pixel greatly improved the discriminative quality of these embeddings and enabled STEm-Seg to achieve state-of-the-art performance on competitive benchmarks with just 3-D or 4-D embeddings, as opposed to the much larger 128-D or 256-D embeddings that are typically used for clustering in ReID tasks.

Chapter 5: In this chapter, we introduced a network architecture called HODOR for tackling Video Object Segmentation (VOS) that could be trained in a weakly supervised manner using only annotated image data. To achieve this, HODOR adopted a transformer-based, encoder-decoder architecture: the encoder was trained to summarize objects given by their respective segmentation masks into concise, high-level 'object descriptors'. The decoder performed the dual function of predicting segmentation masks for the objects given their descriptors. Thus, object masks could be propagated from the first frame throughout the video by applying the encoder and decoder in an alternat-

ing fashion. This approach differs from existing VOS approaches which overwhelmingly adopt the space-time correspondence paradigm which in turn requires densely annotated video data for training. By contrast, the HODOR architecture involves an information bottleneck that enables us to train on static images during training and infer on video data. Moreover, it also allows us to train on video data with temporally sparse labels in a cyclic consistency-based fashion.

Chapter 6: Here we proposed a dataset and associated benchmark called BURST which comprised multiple tasks related to object segmentation in video. This was motivated by the fact that there are currently several benchmarks that evaluate different, but conceptually overlapping video segmentation tasks. However, these benchmarks use different data and evaluation measures. We developed BURST to unify these tasks under a single umbrella with shared data and evaluation measures, thereby facilitating quantitative comparison between tasks and encouraging the development of methods with multi-task capability. BURST was based on the existing TAO dataset for bounding-box level tracking and contains a diverse collection of 3000 video sequences, each 30s long on average, and object annotations belonging to a large set of 482 categories.

Chapter 7: In the second-to-last chapter of this thesis, we proposed TarViS, a universal network architecture that can tackle any task that requires segmenting a set of discrete targets in video. Building upon the motivation for the BURST dataset/benchmark in Chapter 6, we noted that current video segmentation network architectures are largely task-specific even though the set of tasks that they tackle are conceptually overlapping. To enable multi-task capability, TarViS adopted an architecture where the task-specific segmentation targets were modeled as queries which were dynamic inputs to the network. This formulation allowed the core network architecture to be task-agnostic, thus allowing the model to be seamlessly trained and applied to multiple tasks by simply hot-swapping the input queries at runtime. We verified the effectiveness of our approach by training a single model for Video Instance Segmentation (VIS), Video Object Segmentation (VOS), Video Panoptic Segmentation (VPS), and the Point Exemplar-guided Tracking (PET) task from our BURST benchmark.

8.1 Perspectives

The projects discussed in the preceding section were undertaken over a period of four years during which time the field of video segmentation experienced tremendous progress. In this section, we offer a high-level, retrospective view of these projects in terms of the impact they had on the domain and how they can be extended and improved in the future.

STEm-Seg (Chapter 4) offered an alternative to the existing tracking-by-detection paradigm by using spatio-temporal embeddings to segment object instances jointly over

space and time. This approach proved to generalize to other modalities as well since it was later extended to point-cloud segmentation by Aygun *et al.* (2021). In a follow-up work (Mahadevan *et al.*, 2020), we also showed that the 3D convolutional architecture used in STEm-Seg was also effective for the task of video saliency segmentation. Although the publication based on STEm-Seg is highly cited, the potential for continued future extension was somewhat reduced by the emergence of transformer-based architectures for video segmentation which have since dominated these benchmarks.

HODOR (Chapter 5) proposed a different paradigm for VOS based on learning object descriptors rather than space-time correspondences. This inspired the development of our subsequent TarViS network architecture (Chapter 7), and also the interactive image segmentation approach by Rana *et al.* (2023). Although HODOR outperformed existing approaches trained on similar image-level data, when training on dense video data, its performance lagged behind state-of-the-art approaches. A potential direction for future work would be to bridge this performance gap by incorporating space-time correspondences into the architecture in a way that does not compromise the information bottleneck that is key to weakly supervised training. Moreover, one could also extend HODOR to a fully unsupervised setting by using randomly sampled patch masks instead of ground-truth labeled object masks.

The BURST benchmark (Chapter 6) and TarViS (Chapter 7) can be seen as closely related projects: BURST is a benchmark suite that unifies multiple video segmentation tasks, whereas TarViS is a network architecture that can be jointly trained and applied to multiple such tasks. Two of the tasks in the BURST benchmark formed the basis for the '2nd workshop on Tracking and its Many Guises' at the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR) 2023. In terms of interesting future directions, TarViS could be extended to perform language-guided object tracking by formulating a given text prompt as a dynamic query input. Moreover, it can also be extended to perform tasks beyond segmenting discrete targets e.g. depth estimation and bounding-box level detection and tracking.

Bibliography

Alayrac, J.-B., Donahue, J., Luc, P., Miech, A., Barr, I., Hasson, Y., Lenc, K., Mensch, A., Millican, K., Reynolds, M., *et al.* (2022). Flamingo: a visual language model for few-shot learning. In *arXiv*.

Anthony Hu, Alex Kendall, R. C. (2019). Learning a spatio-temporal embedding for video instance segmentation. *arxiv*.

Athar, A., Mahadevan, S., Osep, A., Leal-Taixé, L., and Leibe, B. (2020). Stem-seg: Spatio-temporal embeddings for instance segmentation in videos. In *European Conference on Computer Vision*.

Athar, A., Mahadevan, S., Osep, A., Leal-Taixé, L., and Leibe, B. (2021). A single-stage, bottom-up approach for occluded vis using spatio-temporal embeddings. In *IEEE/CVF International Conference on Computer Vision Workshop*.

Athar, A., Luiten, J., Hermans, A., Ramanan, D., and Leibe, B. (2022). Hodor: High-level object descriptors for object re-segmentation in video learned from static images. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition*.

Athar, A., Luiten, J., Voigtlaender, P., Khurana, T., Dave, A., Leibe, B., and Ramanan, D. (2023a). Burst: A benchmark for unifying object recognition, segmentation and tracking in video. In *Winter Conference on Computer Vision*.

Athar, A., Hermans, A., Luiten, J., Ramanan, D., and Leibe, B. (2023b). Tarvis: A unified architecture for target-based video segmentation. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition*.

Avinash Ramakanth, S. and Venkatesh Babu, R. (2014). Seamseg: Video object segmentation using patch seams. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition*.

Aygun, M., Osep, A., Weber, M., Maximov, M., Stachniss, C., Behley, J., and Leal-Taixe, L. (2021). 4d panoptic lidar segmentation. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition*.

Ba, J. L., Kiros, J. R., and Hinton, G. E. (2016). Layer normalization. *arXiv*.

Bello, I., Zoph, B., Vaswani, A., Shlens, J., and Le, Q. V. (2019). Attention augmented convolutional networks. In *IEEE/CVF International Conference on Computer Vision*.

Bergmann, P., Meinhardt, T., and Leal-Taixé, L. (2019). Tracking without bells and whistles. In *IEEE/CVF International Conference on Computer Vision*.

Berman, M. and Blaschko, M. B. (2018). Optimization of the jaccard index for image segmentation with the lovász hinge. *IEEE/CVF Conference on Computer Vision and Pattern Recognition*.

Berman, M., Rannen Triki, A., and Blaschko, M. B. (2018). The lovász-softmax loss: a tractable surrogate for the optimization of the intersection-over-union measure in neural networks. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition*.

Bernardin, K. and Stiefelhagen, R. (2008). Evaluating multiple object tracking performance: The clear mot metrics. In *Journal on Image and Video Processing*.

Bertasius, G. and Torresani, L. (2020). Classifying, segmenting, and tracking object instances in video with mask propagation. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition*.

Bertasius, G., Wang, H., and Torresani, L. (2021). Is space-time attention all you need for video understanding? In *International Conference on Machine Learning*.

Bochinski, E., Eiselein, V., and Sikora, T. (2017). High-speed tracking-by-detection without using image information. In *IEEE International Conference on Advanced Video and Signal Based Surveillance (AVSS)*.

Brox, T. and Malik, J. (2010). Object segmentation by long term analysis of point trajectories. In *European Conference on Computer Vision*.

Butt, A. A. and Collins, R. T. (2013). Multi-target tracking by lagrangian relaxation to min-cost network flow. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition*.

Caelles, S., Maninis, K.-K., Pont-Tuset, J., Leal-Taixé, L., Cremers, D., and Van Gool, L. (2017). One-shot video object segmentation. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition*.

Caelles, S., Montes, A., Maninis, K.-K., Chen, Y., Van Gool, L., Perazzi, F., and Pont-Tuset, J. (2018). The 2018 davis challenge on video object segmentation. *arXiv*.

Caelles, S., Pont-Tuset, J., Perazzi, F., Montes, A., Maninis, K., and Gool, L. V. (2019). The 2019 DAVIS challenge on VOS: unsupervised multi-object segmentation. *arXiv*.

Cao, J., Anwer, R. M., Cholakkal, H., Khan, F. S., Pang, Y., and Shao, L. (2020). Sipmask: Spatial information preservation for fast image and video instance segmentation. In *European Conference on Computer Vision*.

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*.

Caron, M., Touvron, H., Misra, I., Jégou, H., Mairal, J., Bojanowski, P., and Joulin, A. (2021). Emerging Properties in Self-Supervised Vision Transformers. In *IEEE/CVF International Conference on Computer Vision*.

Caruana, R. (1993). Multitask Learning: A Knowledge-Based Source of Inductive Bias. In *International Conference on Machine Learning*.

Chang, J., Wei, D., and Fisher, J. W. (2013). A video representation using temporal superpixels. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition*.

Chang, M.-F., Lambert, J., Sangkloy, P., Singh, J., Bak, S., Hartnett, A., Wang, D., Carr, P., Lucey, S., Ramanan, D., and Hays, J. (2019). Argoverse: 3d tracking and forecasting with rich maps. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition*.

Chen, L., Papandreou, G., Schroff, F., and Adam, H. (2017). Rethinking atrous convolution for semantic image segmentation. *arXiv*.

Chen, X., Girshick, R., He, K., and Dollár, P. (2019). Tensormask: A foundation for dense object segmentation. In *IEEE/CVF International Conference on Computer Vision*.

Chen, Y., Pont-Tuset, J., Montes, A., and Van Gool, L. (2018). Blazingly fast video object segmentation with pixel-wise metric learning. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition*.

Cheng, B., Collins, M. D., Zhu, Y., Liu, T., Huang, T. S., Adam, H., and Chen, L.-C. (2020a). Panoptic-deeplab: A simple, strong, and fast baseline for bottom-up panoptic segmentation. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition*.

Cheng, B., Choudhuri, A., Misra, I., Kirillov, A., Girdhar, R., and Schwing, A. G. (2021a). Mask2Former for Video Instance Segmentation. *arXiv*.

Cheng, B., Misra, I., Schwing, A. G., Kirillov, A., and Girdhar, R. (2022). Masked-attention Mask Transformer for Universal Image Segmentation. *IEEE/CVF Conference on Computer Vision and Pattern Recognition*.

Cheng, H. K. and Schwing, A. G. (2022). Xmem: Long-term video object segmentation with an atkinson-shiffrin memory model. In *European Conference on Computer Vision*.

Cheng, H. K., Chung, J., Tai, Y.-W., and Tang, C.-K. (2020b). Cascadepsp: toward class-agnostic and very high-resolution segmentation via global and local refinement. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition*.

Cheng, H. K., Tai, Y.-W., and Tang, C.-K. (2021b). Modular Interactive Video Object Segmentation: Interaction-to-Mask, Propagation and Difference-Aware Fusion. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition*.

Cheng, H. K., Tai, Y.-W., and Tang, C.-K. (2021c). Rethinking Space-Time Networks with Improved Memory Coverage for Efficient Video Object Segmentation. In *Neural Information Processing Systems*.

Cheng, J., Tsai, Y.-H., Wang, S., and Yang, M.-H. (2017). Segflow: Joint learning for video object segmentation and optical flow. In *IEEE/CVF International Conference on Computer Vision*.

Cheng, J., Tsai, Y.-H., Hung, W.-C., Wang, S., and Yang, M.-H. (2018). Fast and accurate online video object segmentation via tracking parts. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition*.

Cho, D., Hong, S., and amd J. Kim, S. K. (2019). Key instance selection for unsupervised video object segmentation. *IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshop*.

Choi, W. (2015). Near-online multi-target tracking with aggregated local flow descriptor. In *IEEE/CVF International Conference on Computer Vision*.

Comaniciu, D. and Meer, P. (2002). Mean shift: A robust approach toward feature space analysis. *IEEE Transactions on Pattern Analysis and Machine Intelligence*.

Cordts, M., Omran, M., Ramos, S., Rehfeld, T., Enzweiler, M., Benenson, R., Franke, U., Roth, S., and Schiele, B. (2016). The cityscapes dataset for semantic urban scene understanding. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition*.

Dai, J., Qi, H., Xiong, Y., Li, Y., Zhang, G., Hu, H., and Wei, Y. (2017). Deformable convolutional networks. In *IEEE/CVF International Conference on Computer Vision*.

Dai, X., Chen, Y., Yang, J., Zhang, P., Yuan, L., and Zhang, L. (2021). Dynamic DETR: End-to-End Object Detection With Dynamic Attention. In *IEEE/CVF International Conference on Computer Vision*.

Dalal, N. and Triggs, B. (2005). Histograms of oriented gradients for human detection. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition*.

Dave, A., Tokmakov, P., and Ramanan, D. (2019). Towards segmenting everything that moves. *arXiv*.

Dave, A., Khurana, T., Tokmakov, P., Schmid, C., and Ramanan, D. (2020). Tao: A large-scale benchmark for tracking any object. In *European Conference on Computer Vision*.

De Brabandere, B., Neven, D., and Van Gool, L. (2017). Semantic instance segmentation for autonomous driving. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshop*.

Deng, J., Dong, W., Socher, R., Li, L.-J., Li, K., and Fei-Fei, L. (2009). Imagenet: A large-scale hierarchical image database. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition*.

Devlin, J., Chang, M.-W., Lee, K., and Toutanova, K. (2019). Bert: Pre-training of deep bidirectional transformers for language understanding. *North American Chapter of the Association for Computational Linguistics*.

Dicle, C., Camps, O. I., and Sznaier, M. (2013). The way they move: Tracking multiple targets with similar appearance. In *IEEE/CVF International Conference on Computer Vision*.

Dong, M., Wang, J., Huang, Y., Yu, D., Su, K., Zhou, K., Shao, J., Wen, S., and Wang, C. (2019). Temporal feature augmented network for video instance segmentation. In *IEEE/CVF International Conference on Computer Vision Workshop*.

Dosovitskiy, A., Beyer, L., Kolesnikov, A., Weissenborn, D., Zhai, X., Unterthiner, T., Dehghani, M., Minderer, M., Heigold, G., Gelly, S., *et al.* (2021). An image is worth 16x16 words: Transformers for image recognition at scale. *International Conference on Learning Representations*.

Elgammal, A., Harwood, D., and Davis, L. (2000). Non-parametric model for background subtraction. In *European Conference on Computer Vision*.

Ess, A., Leibe, B., Schindler, K., and Van Gool, L. (2009). Moving obstacle detection in highly dynamic scenes. In *IEEE International Conference on Robotics and Automation*.

Ester, M., Kriegel, H.-P., Sander, J., Xu, X., *et al.* (1996). A density-based algorithm for discovering clusters in large spatial databases with noise. In *Knowledge Discovery and Data Mining*.

Everingham, M., Van Gool, L., Williams, C., Winn, J., and Zisserman, A. (2010). The pascal visual object classes (VOC) challenge. *International Journal of Computer Vision*.

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 *IEEE/CVF Conference on Computer Vision and Pattern Recognition*.

Fan, Q., Zhong, F., Lischinski, D., Cohen-Or, D., and Chen, B. (2015). Jumpcut: non-successive mask transfer and interpolation for video cutout. In *ACM Trans. Graph.*

Feichtenhofer, C., Pinz, A., and Zisserman, A. (2017). Detect to track and track to detect. In *IEEE/CVF International Conference on Computer Vision*.

Felzenszwalb, P. F., Girshick, R. B., McAllester, D., and Ramanan, D. (2009). Object detection with discriminatively trained part-based models. *IEEE Transactions on Pattern Analysis and Machine Intelligence*.

Feng, Q., Yang, Z., Li, P., Wei, Y., and Yang, Y. (2019). Dual embedding learning for video instance segmentation. In *IEEE/CVF International Conference on Computer Vision Workshop*.

Fu, Y., Yang, L., Liu, D., Huang, T. S., and Shi, H. (2021). Compfeat: Comprehensive feature aggregation for video instance segmentation. In *Association for the Advancement of Artificial Intelligence*.

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

Geiger, A., Lenz, P., and Urtasun, R. (2012). Are we ready for autonomous driving? the KITTI vision benchmark suite. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition*.

Ghiasi, G., Zoph, B., Cubuk, E. D., Le, Q. V., and Lin, T.-Y. (2021). Multi-Task Self-Training for Learning General Representations. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition*.

Gkioxari, G. and Malik, J. (2015). Finding action tubes. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition*.

Gori, M., Monfardini, G., and Scarselli, F. (2005). A new model for learning in graph domains. In *IEEE International Joint Conference on Neural Networks*.

Grabner, H., Grabner, M., and Bischof, H. (2006). Real-time tracking via on-line boosting. In *British Machine Vision Conference*.

Grundmann, M., Kwatra, V., Han, M., and Essa, I. (2010). Efficient hierarchical graph-based video segmentation. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition*.

Gu, C., Sun, C., Ross, D. A., Vondrick, C., Pantofaru, C., Li, Y., Vijayanarasimhan, S., Toderici, G., Ricco, S., Sukthankar, R., *et al.* (2018). Ava: A video dataset of spatio-temporally localized atomic visual actions. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition*.

Gupta, A., Dollar, P., and Girshick, R. (2019). LVIS: A dataset for large vocabulary instance segmentation. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition*.

Han, S. H., Hwang, S., Oh, S. W., Park, Y., Kim, H., Kim, M.-J., and Kim, S. J. (2022). Visolo: Grid-based space-time aggregation for efficient online video instance segmentation. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition*.

Han, W., Khorrami, P., Paine, T. L., Ramachandran, P., Babaeizadeh, M., Shi, H., Li, J., Yan, S., and Huang, T. S. (2016). Seq-nms for video object detection. *arXiv*.

Hara, K., Kataoka, H., and Satoh, Y. (2017). Learning spatio-temporal features with 3d residual networks for action recognition. In *ICCV-W*.

He, K., Zhang, X., Ren, S., and Sun, J. (2016). Deep residual learning for image recognition. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition*.

He, K., Gkioxari, G., Dollár, P., and Girshick, R. (2017). Mask R-CNN. In *IEEE/CVF International Conference on Computer Vision*.

Heo, M., Hwang, S., Oh, S. W., Lee, J.-Y., and Kim, S. J. (2022). Vita: Video instance segmentation via object token association. In *Neural Information Processing Systems*.

Hermans, A., Beyer, L., and Leibe, B. (2017). In defense of the triplet loss for person re-identification. *arXiv*.

Homayounfar, N., Liang, J., Ma, W.-C., and Urtasun, R. (2021). Videoclick: Video object segmentation with a single click. *arXiv*.

Hou, R., Chen, C., and Shah, M. (2017). Tube convolutional neural network (t-cnn) for action detection in videos. In *IEEE/CVF International Conference on Computer Vision*.

Hou, R., Chen, C., Sukthankar, R., and Shah, M. (2019). An efficient 3d CNN for action/object segmentation in video. In *British Machine Vision Conference*.

Hu, Y., Huang, J., and Schwing, A. (2017). Maskrnn: Instance level video object segmentation. In *Neural Information Processing Systems*.

Hu, Y.-T., Huang, J.-B., and Schwing, A. G. (2018). Videomatch: Matching based video object segmentation. In *European Conference on Computer Vision*.

Huang, C., Wu, B., and Nevatia, R. (2008). Robust object tracking by hierarchical association of detection responses. In *European Conference on Computer Vision*.

Huang, D.-A., Yu, Z., and Anandkumar, A. (2022). Minvis: A minimal video instance segmentation framework without video-based training. In *Neural Information Processing Systems*.

Hwang, S., Heo, M., Oh, S. W., and Kim, S. J. (2021). Video instance segmentation using inter-frame communication transformers. In *Neural Information Processing Systems*.

Jabri, A., Owens, A., and Efros, A. A. (2020). Space-Time Correspondence as a Contrastive Random Walk. In *Neural Information Processing Systems*.

Jaegle, A., Gimeno, F., Brock, A., Vinyals, O., Zisserman, A., and Carreira, J. (2021). Perceiver: General perception with iterative attention. In *International Conference on Machine Learning*.

Jaegle, A., Borgeaud, S., Alayrac, J.-B., Doersch, C., Ionescu, C., Ding, D., Koppula, S., Zoran, D., Brock, A., Shelhamer, E., *et al.* (2022). Perceiver io: A general architecture for structured inputs & outputs. In *International Conference on Learning Representations*.

Jain, R. and Nagel, H.-H. (1979). On the analysis of accumulative difference pictures from image sequences of real world scenes. *IEEE Transactions on Pattern Analysis and Machine Intelligence*.

Jain, S., Xiong, B., and Grauman, K. (2017). Fusionseg: Learning to combine motion and appearance for fully automatic segmention of generic objects in videos. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition*.

Ji, S., Xu, W., Yang, M., and Yu, K. (2012). 3d convolutional neural networks for human action recognition. In *IEEE Transactions on Pattern Analysis and Machine Intelligence*.

Jonathon Luiten, A. H. (2020). Trackeval. https://github.com/JonathonLuiten/TrackEval.

Kang, K., Li, H., Xiao, T., Ouyang, W., Yan, J., Liu, X., and Wang, X. (2017). Object detection in videos with tubelet proposal networks. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition*.

Kass, M., Witkin, A., and Terzopoulos, D. (1988). Snakes: Active contour models. In *International Journal of Computer Vision*.

Kay, W., Carreira, J., Simonyan, K., Zhang, B., Hillier, C., Vijayanarasimhan, S., Viola, F., Green, T., Back, T., Natsev, P., *et al.* (2017). The kinetics human action video dataset. *arXiv*.

Ke, L., Li, X., Danelljan, M., Tai, Y.-W., Tang, C.-K., and Yu, F. (2021). Prototypical cross-attention networks for multiple object tracking and segmentation. In *Neural Information Processing Systems*.

Khoreva, A., Benenson, R., Ilg, E., Brox, T., and Schiele, B. (2019). Lucid data dreaming for video object segmentation. *International Journal of Computer Vision*.

Kim, D., Woo, S., Lee, J.-Y., and Kweon, I. S. (2020). Video panoptic segmentation. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition*.

Kirillov, A., Girshick, R., He, K., and Dollár, P. (2019). Panoptic feature pyramid networks. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition*.

Kirillov, A., Wu, Y., He, K., and Girshick, R. (2020). Pointrend: Image segmentation as rendering. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition*.

Kokkinos, I. (2017). Ubernet: Training a universal convolutional neural network for low-, mid-, and high-level vision using diverse datasets and limited memory. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition*.

Kolesnikov, A., Pinto, A. S., Beyer, L., Zhai, X., Harmsen, J., and Houlsby, N. (2022). Uvim: A unified modeling approach for vision with learned guiding codes. In *Neural Information Processing Systems*.

Koller, D., Weber, J., and Malik, J. (1994). Robust multiple car tracking with occlusion reasoning. In *European Conference on Computer Vision*.

Kong, S. and Fowlkes, C. C. (2018). Recurrent pixel embedding for instance grouping. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition*.

Kristan, M., Leonardis, A., Matas, J., Felsberg, M., Pflugfelder, R., ˇCehovin Zajc, L., Vojir, T., Bhat, G., Lukezic, A., Eldesokey, A., *et al.* (2018). The sixth visual object tracking vot2018 challenge results. In *European Conference on Computer Vision*.

Krizhevsky, A., Sutskever, I., and Hinton, G. E. (2012). Imagenet classification with deep convolutional neural networks. *Neural Information Processing Systems*.

Kuhn, H. W. and Yaw, B. (1955). The hungarian method for the assignment problem. *Naval Res. Logist. Quart*.

Kuznetsova, A., Rom, H., Alldrin, N., Uijlings, J., Krasin, I., Pont-Tuset, J., Kamali, S., Popov, S., Malloci, M., Kolesnikov, A., *et al.* (2020). The open images dataset v4. *International Journal of Computer Vision*.

Kwak, S., Cho, M., Laptev, I., Ponce, J., and Schmid, C. (2015). Unsupervised object discovery and tracking in video collections. In *IEEE/CVF International Conference on Computer Vision*.

Lai, Z., Lu, E., and Xie, W. (2020). MAST: A Memory-Augmented Self-supervised Tracker. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition*.

Leal-Taixé, L., Milan, A., Reid, I., , Roth, S., and Schindler, K. (2015). MOTChallenge 2015: Towards a benchmark for multi-target tracking. *arXiv*.

Leibe, B., Schindler, K., Cornelis, N., and Gool, L. V. (2008a). Coupled object detection and tracking from static cameras and moving vehicles. *IEEE Transactions on Pattern Analysis and Machine Intelligence*.

Leibe, B., Leonardis, A., and Schiele, B. (2008b). Robust object detection with interleaved categorization and segmentation. *International Journal of Computer Vision*.

Li, F., Kim, T., Humayun, A., Tsai, D., and Rehg, J. M. (2013). Video segmentation by tracking many figure-ground segments. In *IEEE/CVF International Conference on Computer Vision*.

Li, L. H., Zhang, P., Zhang, H., Yang, J., Li, C., Zhong, Y., Wang, L., Yuan, L., Zhang, L., Hwang, J.-N., *et al.* (2022). Grounded language-image pre-training. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition*.

Li, S., Seybold, B., Vorobyov, A., Fathi, A., Huang, Q., and Kuo, C.-C. J. (2018). Instance embedding transfer to unsupervised video object segmentation. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition*.

Li, X. and Loy, C. C. (2018). Video object segmentation with joint re-identification and attention-aware mask propagation. In *European Conference on Computer Vision*.

Li, X., Wei, T., Chen, Y. P., Tai, Y.-W., and Tang, C.-K. (2020). Fss-1000: A 1000-class dataset for few-shot segmentation. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition*.

Liang, S., Shen, X., Huang, J., and Hua, X.-S. (2021). Video Object Segmentation With Dynamic Memory Networks and Adaptive Object Alignment. In *IEEE/CVF International Conference on Computer Vision*.

Liang, Y., Li, X., Jafari, N., and Chen, J. (2020). Video Object Segmentation with Adaptive Feature Bank and Uncertain-Region Refinement. In *Neural Information Processing Systems*.

Lin, H., Wu, R., Liu, S., Lu, J., and Jia, J. (2021). Video instance segmentation with a propose-reduce paradigm. In *IEEE/CVF International Conference on Computer Vision*.

Lin, T.-Y., Maire, M., Belongie, S., Hays, J., Perona, P., Ramanan, D., Dollár, P., and Zitnick, C. L. (2014). Microsoft COCO: Common objects in context. In *European Conference on Computer Vision*.

Lin, T.-Y., Dollár, P., Girshick, R., He, K., Hariharan, B., and Belongie, S. (2017). Feature Pyramid Networks for Object Detection. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition*.

Liu, R., Lehman, J., Molino, P., Such, F. P., Frank, E., Sergeev, A., and Yosinski, J. (2018). An intriguing failing of convolutional neural networks and the coordconv solution. In *Neural Information Processing Systems*.

Liu, X. and Ye, T. (2019). Spatio-temporal attention network for video instance segmentation. In *IEEE/CVF International Conference on Computer Vision Workshop*.

Liu, Y., Zulfikar, I. E., Luiten, J., Dave, A., Ramanan, D., Leibe, B., Osep, A., and Leal-Taixé, L. (2022). Opening up open world tracking. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition*.

Liu, Z., Lin, Y., Cao, Y., Hu, H., Wei, Y., Zhang, Z., Lin, S., and Guo, B. (2021). Swin transformer: Hierarchical vision transformer using shifted windows. In *IEEE/CVF International Conference on Computer Vision*.

Lloyd, S. (1982). Least squares quantization in pcm. *IEEE transactions on information theory*.

Loshchilov, I. and Hutter, F. (2019). Decoupled Weight Decay Regularization. In *International Conference on Learning Representations*.

Lu, X., Wang, W., Danelljan, M., Zhou, T., Shen, J., and Van Gool, L. (2020). Video Object Segmentation with Episodic Graph Memory Networks. In *European Conference on Computer Vision*.

Luiten, J., Voigtlaender, P., and Leibe, B. (2018). Premvos: Proposal-generation, refinement and merging for video object segmentation. In *Asian Conference on Computer Vision*.

Luiten, J., Osep, A., Dendorfer, P., Torr, P., Geiger, A., Leal-Taixé, L., and Leibe, B. (2020). Hota: A higher order metric for evaluating multi-object tracking. *International Journal of Computer Vision*.

Mahadevan, S., Athar, A., Osep, A., Hennen, S., Leal-Taixé, L., and Leibe, B. (2020). Making a case for 3d convolutions for object segmentation in videos. *British Machine Vision Conference*.

Maninis, K.-K., Caelles, S., Chen, Y., Pont-Tuset, J., Leal-Taixé, L., Cremers, D., and Van Gool, L. (2018). Video Object Segmentation Without Temporal Information. *IEEE Transactions on Pattern Analysis and Machine Intelligence*.

McInnes, L., Healy, J., and Astels, S. (2017). hdbscan: Hierarchical density based clustering. *Journal of Open Source Software*, **2**(11), 205.

Meinhardt, T. and Leal-Taixé, L. (2020). Make One-Shot Video Object Segmentation Efficient Again. In *Neural Information Processing Systems*.

Miao, B., Bennamoun, M., Gao, Y., and Mian, A. (2021). Self-Supervised Video Object Segmentation by Motion-Aware Mask Propagation. *arXiv*.

Miao, J., Wang, X., Wu, Y., Li, W., Zhang, X., Wei, Y., and Yang, Y. (2022). Large-scale video panoptic segmentation in the wild: A benchmark. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition*.

Milan, A., Roth, S., and Schindler, K. (2013). Continuous energy minimization for multitarget tracking. *IEEE Transactions on Pattern Analysis and Machine Intelligence*.

Milan, A., Leal-Taixé, L., Schindler, K., and Reid, I. (2015). Joint tracking and segmentation of multiple targets. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition*.

Milan, A., Leal-Taixé, L., Reid, I., Roth, S., and Schindler, K. (2016). MOT16: A benchmark for multi-object tracking. *arXiv*.

Misra, I., Girdhar, R., and Joulin, A. (2021). An end-to-end transformer model for 3d object detection. In *IEEE/CVF International Conference on Computer Vision*.

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 *European Conference on Computer Vision*.

Neuhold, G., Ollmann, T., Bulo, S. R., and Kontschieder, P. (2017). The Mapillary Vistas dataset for semantic understanding of street scenes. In *IEEE/CVF International Conference on Computer Vision*.

Neven, D., Brabandere, B. D., Proesmans, M., and Gool, L. V. (2019). Instance segmentation by jointly optimizing spatial embeddings and clustering bandwidth. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition*.

Newell, A., Huang, Z., and Deng, J. (2017). Associative embedding: End-to-end learning for joint detection and grouping. In *Neural Information Processing Systems*.

Novotny, D., Albanie, S., Larlus, D., and Vedaldi, A. (2018). Semi-convolutional operators for instance segmentation. In *European Conference on Computer Vision*.

Ochs, P. and Brox, T. (2012). Higher order motion models and spectral clustering. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition*.

Ochs, P., Malik, J., and Brox, T. (2013). Segmentation of moving objects by long term video analysis. In *IEEE Transactions on Pattern Analysis and Machine Intelligence*.

Oh, S. W., Lee, J.-Y., Xu, N., and Kim, S. J. (2019). Video Object Segmentation using Space-Time Memory Networks. In *IEEE/CVF International Conference on Computer Vision*.

Okuma, K., Taleghani, A., De Freitas, N., Little, J. J., and Lowe, D. G. (2004). A boosted particle filter: Multitarget detection and tracking. In *European Conference on Computer Vision*.

Osep, A., Mehner, W., Voigtlaender, P., and Leibe, B. (2018). Track, then decide: Category-agnostic vision-based multi-object tracking. *IEEE International Conference on Robotics and Automation*.

Osep, A., Voigtlaender, P., Luiten, J., Breuers, S., and Leibe, B. (2019). Large-scale object mining for object discovery from unlabeled video. In *IEEE International Conference on Robotics and Automation*.

Osep, A., Voigtlaender, P., Weber, M., Luiten, J., and Leibe, B. (2020). 4d generic video object proposals. In *IEEE International Conference on Robotics and Automation*.

Palmer, S. E. (2002). Organizing objects and scenes. *Foundations of cognitive psychology: Core readings*.

Palou, G. and Salembier, P. (2013). Hierarchical video representation with trajectory binary partition tree. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition*.

Paragios, N. and Deriche, R. (2000). Geodesic active contours and level sets for the detection and tracking of moving objects. *IEEE Transactions on Pattern Analysis and Machine Intelligence*.

Parmar, N., Vaswani, A., Uszkoreit, J., Kaiser, L., Shazeer, N., Ku, A., and Tran, D. (2018). Image transformer. In *International Conference on Machine Learning*.

Perazzi, F., Wang, O., Gross, M., and Sorkine-Hornung, A. (2015). Fully connected object proposals for video segmentation. In *IEEE/CVF International Conference on Computer Vision*.

Perazzi, F., Khoreva, A., Benenson, R., Schiele, B., and Sorkine-Hornung, A. (2017). Learning Video Object Segmentation from Static Images. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition*.

Pfeiffer, K., Hermans, A., Sárándi, I., Weber, M., and Leibe, B. (2019). Visual person understanding through multi-task and multi-dataset learning. In *DAGM German Conference on Pattern Recognition*.

Pinheiro, P., Collobert, R., and Dollár, P. (2015). Learning to segment object candidates. In *Neural Information Processing Systems*.

Pinheiro, P., Lin, T., Collobert, R., and Dollár, P. (2016). Learning to refine object segments. In *European Conference on Computer Vision*.

Pirsiavash, H., Ramanan, D., and C.Fowlkes, C. (2011). Globally-optimal greedy algorithms for tracking a variable number of objects. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition*.

Pont-Tuset, J., Perazzi, F., Caelles, S., Arbeláez, P., Sorkine-Hornung, A., and Gool, L. V. (2016). A benchmark dataset and evaluation methodology for video object segmentation. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition*.

Pont-Tuset, J., Perazzi, F., Caelles, S., Arbeláez, P., Sorkine-Hornung, A., and Van Gool, L. (2017). The 2017 DAVIS challenge on video object segmentation. *arXiv*.

Press, O., Smith, N. A., and Lewis, M. (2021). Train Short, Test Long: Attention with Linear Biases Enables Input Length Extrapolation. *arXiv*.

Qi, C. R., Litany, O., He, K., and Guibas, L. J. (2019). Deep hough voting for 3d object detection in point clouds. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition*.

Qi, J., Gao, Y., Hu, Y., Wang, X., Liu, X., Bai, X., Belongie, S., Yuille, A., Torr, P. H., and Bai, S. (2022). Occluded video instance segmentation: A benchmark. In *International Journal of Computer Vision*.

Qiao, S., Zhu, Y., Adam, H., Yuille, A., and Chen, L.-C. (2021). Vip-deeplab: Learning visual perception with depth-aware video panoptic segmentation. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition*.

Radford, A., Kim, J. W., Hallacy, C., Ramesh, A., Goh, G., Agarwal, S., Sastry, G., Askell, A., Mishkin, P., Clark, J., *et al.* (2021). Learning transferable visual models from natural language supervision. In *International Conference on Machine Learning*.

Ramesh, A., Pavlov, M., Goh, G., Gray, S., Voss, C., Radford, A., Chen, M., and Sutskever, I. (2021). Zero-shot text-to-image generation. In *International Conference on Machine Learning*.

Rana, A. K., Mahadevan, S., Hermans, A., and Leibe, B. (2023). Dynamite: Dynamic query bootstrapping for multi-object interactive segmentation transformer. In *IEEE/CVF International Conference on Computer Vision*.

Redmon, J., Divvala, S., Girshick, R., and Farhadi, A. (2016). You only look once: Unified, real-time object detection. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition*.

Reed, S., Zolna, K., Parisotto, E., Colmenarejo, S. G., Novikov, A., Barth-Maron, G., Gimenez, M., Sulsky, Y., Kay, J., Springenberg, J. T., *et al.* (2022). A generalist agent. In *arXiv*.

Ren, S., He, K., Girshick, R., and Sun, J. (2015). Faster R-CNN: Towards real-time object detection with region proposal networks. In *Neural Information Processing Systems*.

Rezatofighi, S. H., Milan, A., Zhang, Z., Shi, Q., Dick, A., and Reid, I. (2015). Joint probabilistic data association revisited. In *IEEE/CVF International Conference on Computer Vision*.

Ristani, E., Solera, F., Zou, R., Cucchiara, R., and Tomasi, C. (2016). Performance measures and a data set for multi-target, multi-camera tracking. In *European Conference on Computer Vision*.

Ruder, S. (2017). An overview of multi-task learning in deep neural networks. *arXiv*.

Sadeghian, A., Alahi, A., and Savarese, S. (2017). Tracking the untrackable: Learning to track multiple cues with long-term dependencies. In *IEEE/CVF International Conference on Computer Vision*.

Seong, H., Hyun, J., and Kim, E. (2020). Kernelized Memory Network for Video Object Segmentation. In *European Conference on Computer Vision*.

Seong, H., Oh, S. W., Lee, J.-Y., Lee, S., Lee, S., and Kim, E. (2021). Hierarchical Memory Matching Network for Video Object Segmentation. In *IEEE/CVF International Conference on Computer Vision*.

Shi, J., Yan, Q., Xu, L., and Jia, J. (2015). Hierarchical image saliency detection on extended cssd. *IEEE Transactions on Pattern Analysis and Machine Intelligence*.

Siam, M., Jiang, C., Lu, S. W., Petrich, L., Gamal, M., Elhoseiny, M., and Jägersand, M. (2018). Video segmentation using teacher-student adaptation in a human robot interaction (HRI) setting. In *IEEE International Conference on Robotics and Automation*.

Sigurdsson, G. A., Varol, G., Wang, X., Farhadi, A., Laptev, I., and Gupta, A. K. (2016). Hollywood in homes: Crowdsourcing data collection for activity understanding. *European Conference on Computer Vision*.

Simonyan, K. and Zisserman, A. (2015). Very deep convolutional networks for large-scale image recognition. In *International Conference on Learning Representations*.

Song, H., Wang, W., Zhao, S., Shen, J., and Lam, K.-M. (2018). Pyramid dilated deeper convlstm for video salient object detection. In *European Conference on Computer Vision*.

Stauffer, C. and Grimson, W. (1999). Adaptive background mixture models for real-time tracking. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition*.

Stauffer, C. and Grimson, W. E. L. (2000). Learning patterns of activity using real-time tracking. *IEEE Transactions on Pattern Analysis and Machine Intelligence*.

Stiefelhagen, R., Bernardin, K., Bowers, R., Rose, R. T., Michel, M., and Garofolo, J. (2007). The clear 2007 evaluation. In *Multimodal Technologies for Perception of Humans*.

Sun, D., Yang, X., Liu, M.-Y., and Kautz, J. (2018). PWC-Net: CNNs for optical flow using pyramid, warping, and cost volume. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition*.

Sun, P., Kretzschmar, H., Dotiwalla, X., Chouard, A., Patnaik, V., Tsui, P., Guo, J., Zhou, Y., Chai, Y., Caine, B., Vasudevan, V., Han, W., Ngiam, J., Zhao, H., Timofeev, A., Ettinger, S., Krivokon, M., Gao, A., Joshi, A., Zhang, Y., Shlens, J., Chen, Z., and Anguelov, D. (2020). Scalability in perception for autonomous driving: Waymo open dataset. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition*.

Sundaram, N., Brox, T., and Keutzer, K. (2010). Dense point trajectories by gpu-accelerated large displacement optical flow. In *European Conference on Computer Vision*.

Tang, S., Andriluka, M., Andres, B., and Schiele, B. (2017). Multiple people tracking by lifted multicut and person re-identification. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition*.

Teichman, A., Levinson, J., and Thrun, S. (2011). Towards 3D object recognition via classification of arbitrary object tracks. In *IEEE International Conference on Robotics and Automation*.

Thomee, B., Shamma, D. A., Friedland, G., Elizalde, B., Ni, K., Poland, D., Borth, D., and Li, L.-J. (2016). Yfcc100m: The new data in multimedia research. *Communications of the ACM*.

Tokmakov, P., Alahari, K., and Schmid, C. (2017). Learning video object segmentation with visual memory. In *IEEE/CVF International Conference on Computer Vision*.

Tokmakov, P., Li, J., and Gaidon, A. (2023). Breaking the" object" in video object segmentation. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition*.

Tran, D., Bourdev, L., Fergus, R., Torresani, L., and Paluri, M. (2015). Learning spatiotemporal features with 3d convolutional networks. In *IEEE/CVF International Conference on Computer Vision*.

Tran, D., Wang, H., Torresani, L., Ray, J., LeCun, Y., and Paluri, M. (2018). A closer look at spatiotemporal convolutions for action recognition. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition*.

Tran, D., Wang, H., Torresani, L., and Feiszli, M. (2019). Video classification with channel-separated convolutional networks. In *IEEE/CVF International Conference on Computer Vision*.

Tsai, D., Flagg, M., Nakazawa, A., and Rehg, J. M. (2012). Motion coherent tracking using multi-label mrf optimization. In *International Journal of Computer Vision*.

Uijlings, J. R. R., van de Sande, K. E. A., Gevers, T., and Smeulders, A. W. M. (2013). Selective search for object recognition. *International Journal of Computer Vision*.

Van den Bergh, M., Roig, G., Boix, X., Manen, S., and Van Gool, L. (2013). Online video seeds for temporal window objectness. In *IEEE/CVF International Conference on Computer Vision*.

Varol, G., Laptev, I., and Schmid, C. (2017). Long-term temporal convolutions for action recognition. In *IEEE Transactions on Pattern Analysis and Machine Intelligence*.

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, Ł., and Polosukhin, I. (2017). Attention Is All You Need. In *Neural Information Processing Systems*.

Ventura, C., Bellver, M., Girbau, A., Salvador, A., Marqués, F., and Gir'o i Nieto, X. (2019). RVOS: end-to-end recurrent network for video object segmentation. *IEEE/CVF Conference on Computer Vision and Pattern Recognition*.

Voigtlaender, P. and Leibe, B. (2017). Online adaptation of convolutional neural networks for video object segmentation. In *British Machine Vision Conference*.

Voigtlaender, P., Chai, Y., Schroff, F., Adam, H., Leibe, B., and Chen., L.-C. (2019a). Feelvos: Fast end-to-end embedding learning for video object segmentation. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition*.

Voigtlaender, P., Krause, M., Osep, A., Luiten, J., Sekar, B., Geiger, A., and Leibe, B. (2019b). MOTS: Multi-object tracking and segmentation. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition*.

Voigtlaender, P., Luiten, J., Torr, P. H., and Leibe, B. (2020). Siam R-CNN: Visual Tracking by Re-Detection. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition*.

Voigtlaender, P., Luo, L., Yuan, C., Jiang, Y., and Leibe, B. (2021). Reducing the annotation effort for video object segmentation datasets. In *Winter Conference on Computer Vision*.

Vondrick, C., Shrivastava, A., Fathi, A., Guadarrama, S., and Murphy, K. (2018). Tracking emerges by colorizing videos. In *European Conference on Computer Vision*.

Wang, L., Hua, G., Sukthankar, R., Xue, J., and Zheng, N. (2014). Video object discovery and co-segmentation with extremely weak supervision. In *European Conference on Computer Vision*.

Wang, L., Lu, H., Wang, Y., Feng, M., Wang, D., Yin, B., and Ruan, X. (2017). Learning to detect salient objects with image-level supervision. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition*.

Wang, Q., He, Y., Yang, X., Yang, Z., and Torr, P. (2019a). An empirical study of detection-based video instance segmentation. In *IEEE/CVF International Conference on Computer Vision Workshop*.

Wang, Q., Zhang, L., Bertinetto, L., Hu, W., and Torr, P. H. (2019b). Fast Online Object Tracking and Segmentation: A Unifying Approach. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition*.

Wang, W., Shen, J., and Shao, L. (2015). Consistent video saliency using local gradient flow optimization and global refinement. In *Transactions on Image Processing*.

Wang, W., Lu, X., Shen, J., Crandall, D. J., and Shao, L. (2019c). Zero-shot video object segmentation via attentive graph neural networks. In *IEEE/CVF International Conference on Computer Vision*.

Wang, W., Feiszli, M., Wang, H., and Tran, D. (2021a). Unidentified video objects: A benchmark for dense, open-world segmentation. In *IEEE/CVF International Conference on Computer Vision*.

Wang, X., Jabri, A., and Efros, A. A. (2019d). Learning Correspondence from the Cycle-Consistency of Time. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition*.

Wang, X., Zhang, R., Shen, C., Kong, T., and Li, L. (2021b). Solo: A simple framework for instance segmentation. In *IEEE Transactions on Pattern Analysis and Machine Intelligence*.

Wang, Y., Xu, Z., Wang, X., Shen, C., Cheng, B., Shen, H., and Xia, H. (2021c). End-to-end video instance segmentation with transformers. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition*.

Weber, M., Xie, J., Collins, M., Zhu, Y., Voigtlaender, P., Adam, H., Green, B., Geiger, A., Leibe, B., Cremers, D., Osep, A., Leal-Taixé, L., and Chen, L.-C. (2021). STEP: Segmenting and tracking every pixel. In *Neural Information Processing Systems*.

Wojke, N., Bewley, A., and Paulus., D. (2017a). Onboard contextual classification of 3D point clouds with learned high-order markov random fields. In *International Conference on Image Processing*.

Wojke, N., Bewley, A., and Paulus, D. (2017b). Simple online and realtime tracking with a deep association metric. In *International Conference on Image Processing*.

Woo, S., Kim, D., Lee, J.-Y., and Kweon, I. S. (2021). Learning to associate every segment for video panoptic segmentation. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition*.

Wren, C. R., Azarbayejani, A., Darrell, T., and Pentland, A. (1997). Pfinder: Real-time tracking of the human body. *IEEE Transactions on Pattern Analysis and Machine Intelligence*.

Wu, J., Liu, Q., Jiang, Y., Bai, S., Yuille, A., and Bai, X. (2022a). In defense of online models for video instance segmentation. In *European Conference on Computer Vision*.

Wu, J., Jiang, Y., Zhang, W., Bai, X., and Bai, S. (2022b). Seqformer: a frustratingly simple model for video instance segmentation. In *European Conference on Computer Vision*.

Wu, Y. and He, K. (2018). Group normalization. In *European Conference on Computer Vision*.

Wug Oh, S., Lee, J.-Y., Sunkavalli, K., and Joo Kim, S. (2018). Fast video object segmentation by reference-guided mask propagation. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition*.

Xiao, F. and Jae Lee, Y. (2016). Track and segment: An iterative unsupervised approach for video object proposals. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition*.

Xie, H., Yao, H., Zhou, S., Zhang, S., and Sun, W. (2021). Efficient regional memory network for video object segmentation. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition*.

Xu, C. (2012). Evaluation of super-voxel methods for early video processing. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition*.

Xu, N., Yang, L., Fan, Y., Yue, D., Liang, Y., Yang, J., and Huang, T. (2018a). YouTube-VOS: A Large-Scale Video Object Segmentation Benchmark. *arXiv*.

Xu, N., Yang, L., Fan, Y., Yang, J., Yue, D., Liang, Y., Price, B., Cohen, S., and Huang, T. (2018b). YouTube-VOS: Sequence-to-sequence video object segmentation. In *European Conference on Computer Vision*.

Yan, B., Jiang, Y., Sun, P., Wang, D., Yuan, Z., Luo, P., and Lu, H. (2022). Towards grand unification of object tracking. In *European Conference on Computer Vision*.

Yang, L., Wang, Y., Xiong, X., Yang, J., and Katsaggelos, A. K. (2018). Efficient video object segmentation via network modulation. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition*.

Yang, L., Fan, Y., and Xu, N. (2019a). Video instance segmentation. In *IEEE/CVF International Conference on Computer Vision*.

Yang, S., Fang, Y., Wang, X., Li, Y., Fang, C., Shan, Y., Feng, B., and Liu, W. (2021a). Crossover learning for fast online video instance segmentation. In *IEEE/CVF International Conference on Computer Vision*.

Yang, Z., Wang, Q., Bertinetto, L., Hu, W., Bai, S., and Torr, P. H. S. (2019b). Anchor diffusion for unsupervised video object segmentation. In *IEEE/CVF International Conference on Computer Vision*.

Yang, Z., Wei, Y., and Yang, Y. (2020). Collaborative Video Object Segmentation by Foreground-Background Integration. In *European Conference on Computer Vision*.

Yang, Z., Wei, Y., and Yang, Y. (2021b). Associating Objects with Transformers for Video Object Segmentation. In *Neural Information Processing Systems*.

Yeong Jun Koh, C.-S. K. (2017). Primary object segmentation in videos based on region augmentation and reduction. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition*.

Yoon, Y.-C., Boragule, A., Yoon, K., and Jeon, M. (2018). Online multi-object tracking with historical appearance matching and scene adaptive detection filtering. *IEEE International Conference on Advanced Video and Signal Based Surveillance (AVSS)*.

Yu, F., Chen, H., Wang, X., Xian, W., Chen, Y., Liu, F., Madhavan, V., and Darrell, T. (2020). Bdd100k: A diverse driving dataset for heterogeneous multitask learning. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition*.

Yu, J. and Blaschko, M. (2015). Learning submodular losses with the lovász hinge. In *International Conference on Machine Learning*.

Zamir, A. R., Sax, A., Shen, W., Guibas, L. J., Malik, J., and Savarese, S. (2018). Taskonomy: Disentangling task transfer learning. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition*.

Zeng, X., Liao, R., Gu, L., Xiong, Y., Fidler, S., and Urtasun, R. (2019a). Dmm-net: Differentiable mask-matching network for video object segmentation. In *IEEE/CVF International Conference on Computer Vision*.

Zeng, Y., Zhang, P., Zhang, J., Lin, Z., and Lu, H. (2019b). Towards high-resolution salient object detection. In *IEEE/CVF International Conference on Computer Vision*.

Zhao, H., Yan, Z., Torresani, L., and Torralba, A. (2019). Hacs: Human action clips and segments dataset for recognition and temporal localization. *arXiv*.

Zhou, B., Zhao, H., Puig, X., Fidler, S., Barriuso, A., and Torralba, A. (2017). Scene parsing through ade20k dataset. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition*.

Zhu, X., Su, W., Lu, L., Li, B., Wang, X., and Dai, J. (2020). Deformable detr: Deformable transformers for end-to-end object detection. In *International Conference on Learning Representations*.

Zulfikar, I. E., Luiten, J., and Leibe, B. (2019). Unovost: Unsupervised offline video object segmentation and tracking for the 2019 unsupervised davis challenge. *IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshop*.

Curriculum Vitae

Ali Athar

E-Mail	athar@vision.rwth-aachen.de
Homepage	https://www.aliathar.net/
Date of Birth	22.02.1994
Place of Birth	Karachi, Pakistan

Education

Apr. 2019 – Mar. 2024	PhD in Computer Science
Oct. 2016 – Feb. 2019	MSc in Communications Engineering
Sep. 2012 – Jun. 2016	BE in Electrical Engineering

Industry Experience

Feb. 2024 – Present	Research Scientist at ByteDance
Jan. 2023 – Jun. 2023	Reserch Intern at Waabi Innocation Inc.
Feb. 2017 – Feb. 2019	Software Engineer (part-time) at NavVis GmbH.

List of Publications

A. Athar, S. Mahadevan, A. Osep, L. Leal-Taixe, B. Leibe. *STEm-Seg: Spatio-temporal Embeddings for Instance Segmentation in Videos*, European Conference on Computer Vision, 2020

S. Mahadevan, A. Athar, A. Osep, L. Leal-Taixe, B. Leibe. *Making a Case for 3D Convolutions for Object Segmentation in Videos*, British Machine Vision Conference, 2020

C. Schmidt, A. Athar, S. Mahadevan, B. Leibe. *D^2-Conv3D: Dynamic Dilated Convolutions for Object Segmentation in Videos*, Winter Conference on Applications of Computer Vision, 2022

A. Athar, J. Luiten, A. Hermans, D. Ramanan, B. Leibe. *HODOR: High-level Object Descriptors for Object Re-segmentation in Video Learned from Static Images*, IEEE/CVF Conference on Computer Vision and Pattern Recognition, 2022

A. Athar, J. Luiten, P. Voigtlaender, T. Khurana, A. Dave, B. Leibe, D. Ramanan. *BURST: A Benchmark for Unifying Object Recognition, Segmentation and Tracking in Video*, Winter Conference on Applications of Computer Vision, 2023

A. Athar, A. Hermans, J. Luiten, D. Ramanan, B. Leibe. *TarViS: A Unified Approach for Target-based Video Segmentation*, IEEE/CVF Conference on Computer Vision and Pattern Recognition, 2023

A. Athar, E. Li, S. Casas, R. Urtasun. *4DFormer: Multimodal 4D Panoptic Segmentation*, Conference on Robot Learning, 2023