International Journal of Computer Vision
https://doi.org/10.1007/s11263-022-01703-8

Recurrent Graph Neural Networks for Video Instance Segmentation

Emil Brissman[1,4] · **Joakim Johnander**[1,5] · **Martin Danelljan**[3] · **Michael Felsberg**[1,2]

Received: 14 February 2022 / Accepted: 13 October 2022

Abstract
Video instance segmentation is one of the core problems in computer vision. Formulating a purely learning-based method, which models the generic track management required to solve the video instance segmentation task, is a highly challenging problem. In this work, we propose a novel learning framework where the entire video instance segmentation problem is modeled jointly. To this end, we design a graph neural network that in each frame jointly processes all detections and a memory of previously seen tracks. Past information is considered and processed via a recurrent connection. We demonstrate the effectiveness of the proposed approach in comprehensive experiments. Our approach operates online at over 25 FPS and obtains 16.3 AP on the challenging OVIS benchmark, setting a new state-of-the-art. We further conduct detailed ablative experiments that validate the different aspects of our approach. Code is available at https://github.com/emibr948/RGNNVIS-PlusPlus.

Keywords Detection · Tracking · Segmentation · Video

1 Introduction

Video instance segmentation (VIS) is the task of simultaneously detecting, segmenting, and tracking object instances from a set of predefined classes. This task has a wide range of applications in autonomous driving (Cordts et al., 2016; Yu et al., 2020), data annotation (Izquierdo et al., 2019; Berg et al., 2019), and biology (T'Jampens et al., 2016; Zhang et al., 2008; Burghardt & Ćalić, 2006). In contrast to image instance segmentation, the temporal aspect of its video counterpart poses several additional challenges. Preserving correct instance identities across frames is made difficult by the presence of other similar instances. Objects may be subject to occlusions, fast motion, or major appearance changes. Moreover, the videos can be subject to wild camera motion and severe background clutter.

Prior work on video instance segmentation has taken inspiration from related areas of multiple object tracking, video object detection, instance segmentation, and video object segmentation (Yang et al., 2019; Athar et al., 2020; Bertasius & Torresani, 2020). Most methods adopt the tracking-by-detection paradigm popular in multiple object tracking (Brasó & Leal-Taixé, 2020). In this paradigm, an instance segmentation method provides detections in each frame, reducing the task to the formation of *tracks*. Given a set of already initialized tracks, one must determine for each detection whether it belongs to one of the tracks, if it is a false positive, or if it should initialize a new track. Most approaches (Yang et al., 2019; Cao et al., 2020; Bertasius & Torresani, 2020; Luiten et al., 2019) learn to match pairs of detections and then rely on heuristics to form the final output, *e.g.*, initializing new tracks, predicting confidences, removing tracks, and predicting class memberships.

Communicated by Alexander Schwing.

These authors contributed equally to this work.

✉ Joakim Johnander
joakim.johnander@liu.se

Emil Brissman
emil.brissman@liu.se

Martin Danelljan
martin.danelljan@vision.ee.ethz.ch

Michael Felsberg
michael.felsberg@liu.se

[1] Computer Vision Laboratory, Department of Electrical Engineering, Linköping University, Linköping, Sweden

[2] School of Engineering, University of KwaZulu-Natal, Durban, South Africa

[3] Computer Vision Lab, ETH Zürich, Zürich, Switzerland

[4] Saab, Linköping, Sweden

[5] Zenseact, Göteborg, Sweden

Published online: 18 November 2022

The aforementioned pipelines suffer from two major drawbacks. (i) The learnt models lack flexibility, and are for instance unable to reason globally over all detections or access information temporally (Yang et al., 2019; Cao et al., 2020). (ii) The model learning stage does not closely model the inference, for instance by utilizing only pairs of frames or ignoring subsequent detection merging stages (Yang et al., 2019; Cao et al., 2020; Luiten et al., 2019; Bertasius & Torresani, 2020). This means that the method never gets the chance to learn many of the aspects of the VIS problem – such as dealing with false positives in the employed instance segmentation method or handling uncertain detections.

An earlier version of this work addressed these two drawbacks via a spatiotemporal learning framework where the VIS-problem is closely modelled (Johnander et al., 2021). In this framework, a neural network proceeds frame by frame, and is in each frame supposed to create tracks, associate detections to tracks, and score existing tracks. This formulation is used to train a flexible model that in each frame processes all tracks and detections jointly via a graph neural network (GNN), and considers past information via a recurrent connection. The model predicts, for each detection, a probability that the detection should initialize a new track. The model also predicts, for each pair of existing tracks and detections, the probability of instance correspondence. Finally, it predicts an embedding for each existing track. The embedding serves two purposes: (i) it is used to predict confidence and class for the track; and (ii) it is via the recurrent connection fed as input to the GNN in the next frame.

In this work, we analyze the approach for long time videos and heavily crowded scenes. These scenarios are highly challenging due to occlusions, objects going out of view or into view, a moving camera, and the presence of many similar objects near each other. Qi et al. (2021) highlighted these issues and proposed a new benchmark that enables a more detailed analysis of these aspects. We experiment with the aforementioned learning formulation and neural network and find that it generalizes fairly well to this new benchmark. Furthermore, we propose three extensions to the approach that improve the results in those scenarios.

1.1 Contributions

Our main contributions (Johnander et al., 2021) are as follows.

(i) We propose a new framework for training video instance segmentation methods. The methods proceed frame-by-frame and are in each frame – given detections from an instance segmentation network – trained to match detections to tracks, initialize new tracks, predict segmentations, and score tracks.
(ii) We present a suitable and flexible model based on Graph Neural Networks and Recurrent Neural Networks.
(iii) We show that the GNN successfully learns to propagate information between different tracks and detections in order to predict matches, initialize new tracks, and predict track confidence and class.
(iv) A recurrent connection allows us to feed information about the tracks to the next time step. We show that, while a naïve implementation of such a connection leads to highly unstable training, an adaption of the long short-term memory effectively solves this issue.
(v) We model the instance appearance as a Gaussian distribution and introduce a learnable update formulation.
(vi) We analyze the effectiveness of our approach in comprehensive experiments. Our method outperforms previous near real-time approaches with a relative mAP gain of 9.0% on the YouTubeVIS dataset (Yang et al., 2019).

Compared to Johnander et al. (2021), our main contributions are as follows.

(i) We show that the proposed model—based on Graph Neural Networks and Recurrent Neural Networks—generalizes to more challenging data and obtains state-of-the-art performance.
(ii) We introduce a positional embedding, inspired from the literature on transformers (Vaswani et al., 2017; Carion et al., 2020), and show that this aids performance.
(iii) We aim to learn the VIS-problem and hinge on the availability of good data. To this end, we investigate the performance improvements from concatenating data from different benchmarks.

2 Related Work

The video instance segmentation (VIS) problem was introduced by Yang et al. (2019). With it, they proposed several straightforward approaches to tackle the task. They follow the tracking-by-detection paradigm and apply an instance segmentation method to provide detections in each frame, then form tracks based on these detections. Furthermore, Yang et al. (2019) also experiment with several approaches to match detections: mask propagation with a video object segmentation method (Voigtlaender et al., 2019); application of a multiple object tracking method (Wojke et al., 2017), in which the image-plane bounding boxes are Kalman filtered, and targets are re-detected with a learned re-identification mechanism; and finally, similarity learning of instance-specific appearance descriptors (Yang et al., 2019). Additionally, they experiment with the offline temporal filtering proposed in Han et al. (2016).

Li et al. (2021) propose a strategy to align features based on anchor boxes and a temporal fusion strategy in order to better address challenges such as occlusion and motion blur. Qi et al. (2021) propose to handle occlusions with a different temporal fusion strategy. Yang et al. (2021) propose to let the model predict a filter that is used to segment the target, and train the model to, for each object, produce a filter that accurately segments the object in multiple frames. Cao et al. (2020) propose to improve the underlying instance segmentation method, obtaining better performance and computational efficiency. Luiten et al. (2019) propose (i) to improve the instance segmentation method by applying different networks for classification, segmentation, and proposal generation; and (ii) to form tracks with the offline algorithm proposed in Luiten et al. (2020). Bertasius and Torresani (2020) also utilize a more powerful instance segmentation method (Bertasius et al., 2018), and propose a novel mask propagation method based on deformable convolutions. Both (Luiten et al., 2019) and (Bertasius & Torresani, 2020) achieve strong performance, but at a very high computational cost.

These approaches try various ways to improve the underlying instance segmentation method or the association of detections. Works by Yang et al. (2019); Luiten et al. (2019) mostly rely on heuristics and are not end-to-end trainable. Furthermore, the track scoring step, where the class and confidence are predicted, has received little attention and is, in existing approaches, calculated with a majority vote and an averaging operation. Athar et al. (2020) instead propose an end-to-end trainable approach that is trained to predict instance center heatmaps and an embedding for each pixel. A high response in the heatmap represents a track. The track is constructed by matching the embedding at the position of the track response with the embeddings of all other pixels. Each pixel with a sufficiently similar embedding is assigned to the track.

The method proposed by Athar et al. (2020) is extended with a larger feature extraction backbone in Athar et al. (2021) to get higher accuracy. Similarly, Li et al. (2021) use a transformer backbone (Liu et al., 2021) to boost performance. Works by Wang et al. (2021) and Hwang et al. (2021) propose end-to-end trainable frameworks built upon transformers (Vaswani et al., 2017; Carion et al., 2020), and view the video instance segmentation problem as sequence decoding. These approaches obtain good performance but are *offline*, observing the entire video prior to making predictions on any frame.

Our approach is closely related to two works on multiple object tracking (MOT) (Brasó & Leal-Taixé, 2020; Weng et al., 2020) and work on feature matching (Sarlin et al., 2020). These works associate detections or feature points by forming a bipartite graph and applying a Graph Neural Network. The strength of this approach is that the neural network simultaneously reasons about all available information. However, the setting of these works differs significantly from video instance segmentation. MOT is typically restricted to a specific type of scene, such as automotive, and usually with only one or two classes. Furthermore, for both MOT and feature matching, no classification or confidence will be provided for the tracks. This is reflected in the way Brasóand Leal-Taixé (2020), Weng et al. (2020), and Sarlin et al. (2020) utilize their GNNs, where only either nodes or edges are of interest, not both. The other part exists solely for the purpose of passing messages. As we explain in Sect. 3, we will instead utilize both edges and nodes: the edges to predict association and the nodes to predict class membership and confidence.

3 Method

We propose an approach for video instance segmentation, consisting of a single neural network. Our model proceeds frame by frame, and performs the following steps: (i) it predicts tentative single-image instance segmentations, (ii) associate detections to existing tracks, (iii) initialize new tracks, (iv) score existing tracks, and (v) update the states of each track.

The instance segmentations together with the existing tracks are fed into a graph neural network (GNN). The GNN processes all tracks and detections jointly to produce output embeddings that are used for association and scoring. These output embeddings are furthermore fed as input to the GNN in the next time step, enabling the GNN to process both present and previous information. An overview of the approach is provided in Fig. 1.

3.1 Track-Detection Association

We maintain a memory of previously seen objects, or *tracks*, which is updated over time. In each frame, an instance segmentation method produces tentative detections. The aim of our model is to associate detections with tracks, determining whether or not track m corresponds to detection n. In addition, the model needs to decide, for each detection n, whether it should initialize a new track.

3.1.1 Motivation

Most existing methods (Yang et al., 2019; Luiten et al., 2019; Cao et al., 2020) associate tracks to detections by training a network to extract appearance descriptors. The descriptors are trained to be similar if they correspond to the same object, and dissimilar if they correspond to different objects. The issue with such an approach is that appearance descriptors corresponding to visually and semantically similar, but different instances, will be trained to be different. In such sce-

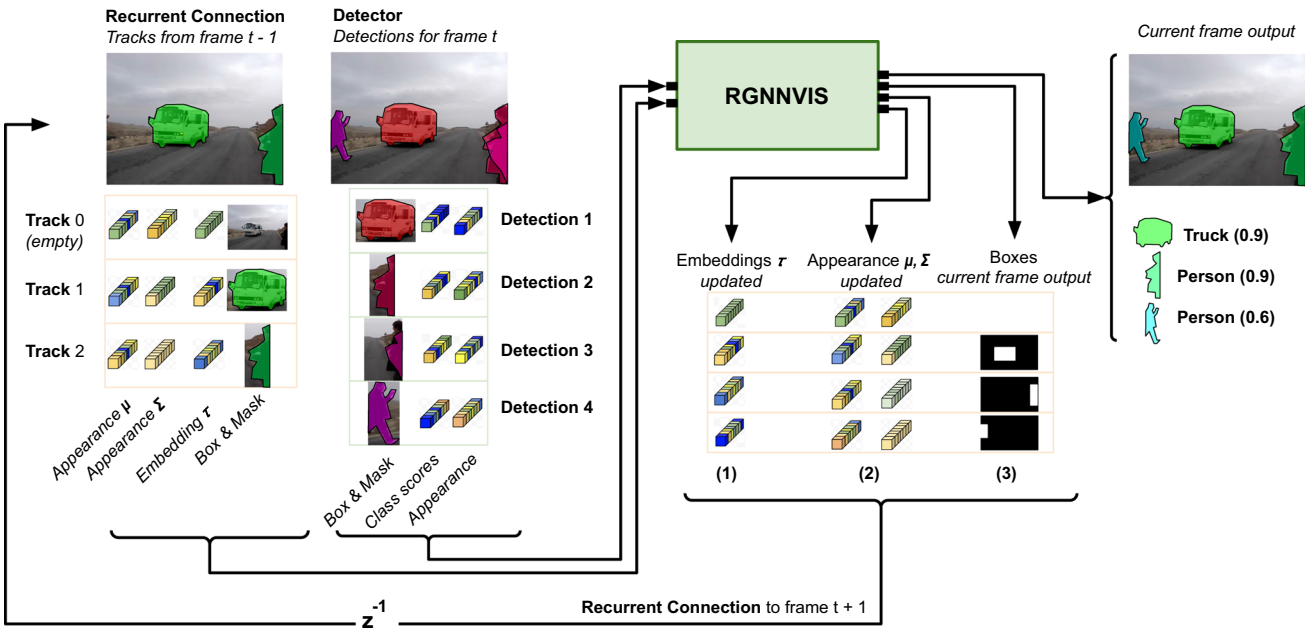

Fig. 1 Overview over the proposed approach. The approach proceeds frame-by-frame and in each frame, a memory of tracks and a set of detections is fed into a recurrent graph neural network (RGNN). Based on the two input sets, the RGNN initializes new tracks and for each initialized track, predicts a segmentation, confidence and class. Furthermore, the RGNN constructs a track embedding and an appearance model for each track, including the newly initialized tracks, which together with the predicted boxes of each track is fed to the next frame via a recurrent connection. See Figs. 2, 3, 4 and 5 for details on the different components of RGNNVIS

narios it might be better to let the appearance descriptors be similar, and instead rely on for instance spatial information. The network should therefore assess all available information before making its decision.

Further information is obtained from track-detection pairs other than the one considered. It may be difficult to determine whether a track and a detection match in isolation, for instance with cluttered scenes or when visibility is poor. In such scenarios, the instance segmentation method might provide multiple detections that all overlap the same object to some extent. Another difficult scenario is when there is sudden and severe camera motion, in which case we might need global reasoning in order to either disregard spatial similarity or treat it differently. We therefore hypothesize that it is important for the network to *reason about all tracks and detections simultaneously*.

The same is true when determining whether a detection should initialize a new track. How well a detection matches existing tracks must influence this decision. Previous works (Yang et al., 2019; Cao et al., 2020) achieve this with a hard decision. In these, a new track will be initialized for each detection that does not match an existing track. We avoid this heuristic and instead let the network process all tracks and detections simultaneously and jointly predict track-detection assignment and track initialization. It should be noted, however, that the detections are noisy in general. Making the correct decision may be outright impossible. In such scenarios we would expect the model to create a track, and over time as more information is accumulated, re-evaluate whether the track is novel, previously seen, or from a false positive in the detector.

3.1.2 Graph Construction

For each detection n we construct an embedding δ_n. It is initialized as the concatenation of the bounding box and classification scores output by the detector. Each track in memory has an embedding τ_m, that has been produced by our model in the previous time step via the *recurrent connection*. We represent the relationship between each track-detection pair with an embedding e_{mn}. This embedding will later be used to predict the probability that track m matches detection n. It is initialized as the concatenation of the spatial similarity and the appearance similarity. The spatial similarity is the Jaccard index between the detection bounding box and the track bounding box predicted in the frame where the track was last seen. The appearance similarity is based on the loglikelihood of the detection appearance, given the track appearance model. It is described in detail in Sect. 3.2. The graph construction is illustrated in Fig. 2. Furthermore, we let the relationship between each detection and a corresponding potential new track be represented with an embedding e_{0n}, and let τ_0 represent an empty track embedding. We treat e_{0n} and τ_0 the way we treat other edges and tracks, but they

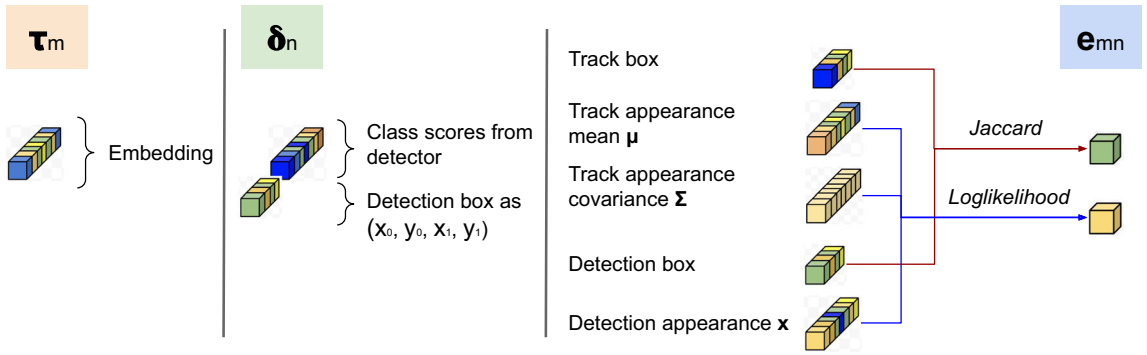

Fig. 2 Overview of the node and edge initialization during graph construction. Each track embedding τ_m was output by our approach in the previous frame. The detection embedding δ_n is constructed from the nth detection provided by the detector in the current frame. The edge embedding e_{mn} is initialized by comparing track m to detection n. The track box, appearance mean, and appearance covariance were output by our approach in the last frame where track m was seen

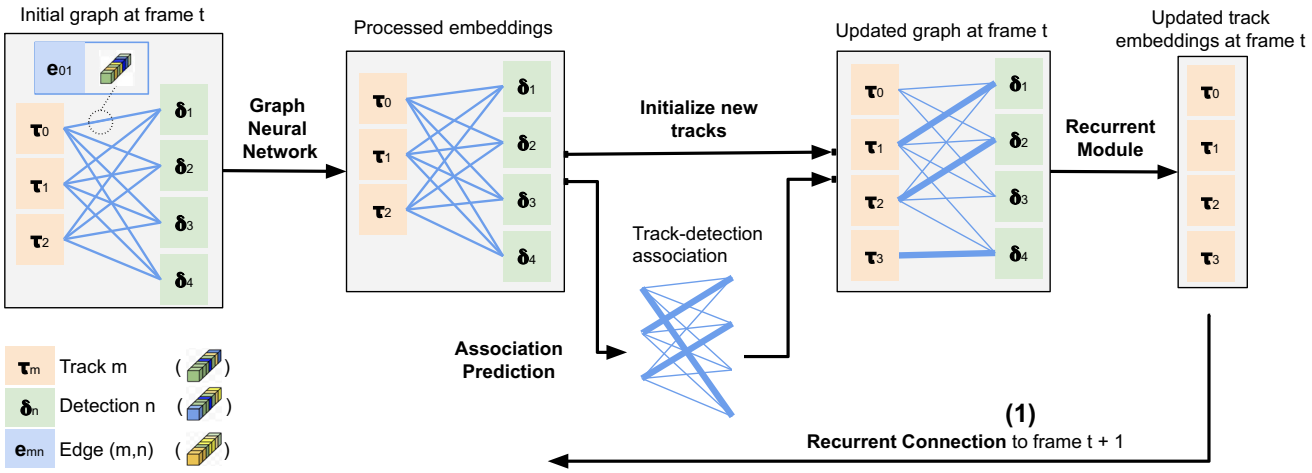

Fig. 3 Illustration of the graph neural network (GNN) within the RGNNVIS-module at a single frame t. The GNN globally processes the track embeddings $\{\tau_m\}_m$, detection embeddings $\{\delta_n\}_n$, and edges $\{e_{mn}\}_{mn}$. The processed edge embeddings are used to associate detections to tracks. The track embedding τ_0 corresponds to an empty track, and its associated detections initialize new tracks, yielding an updated graph. The processed track embeddings are fed through a recurrent module (see Sect. 3.3) and recurrently fed as input to RGNNVIS in the next frame

are processed with their own set of weights. The initialization of the edges e_{0n} is done without the spatial similarity and only with the appearance similarity. We maintain a separate appearance model for the empty track, based on the appearance of the entire scene. The elements τ_m, δ_n, and e_{mn} constitute a bipartite graph, as illustrated in Fig. 3.

$$e_{mn}^{i+1} = f_i^e([e_{mn}^i, \tau_m^i, \delta_n^i]) \ , \tag{1a}$$

$$\tau_m^{i+1} = f_i^\tau([\tau_m^i, \sum_j g_i^\tau(e_{mj}^{i+1}) e_{mj}^{i+1}]) \ , \tag{1b}$$

$$\delta_n^{i+1} = f_i^\delta([\delta_n^i, \sum_j g_i^\delta(e_{jn}^{i+1}) e_{jn}^{i+1}]) \ . \tag{1c}$$

3.1.3 GNN-based Association

The idea is to propagate information between the different embeddings in a learnable way, providing updated embeddings that can directly be used to predict the quantities needed for video instance segmentation. To this end, we adopt layers that perform updates of the form

Here, i enumerates the network layers. Each function f_i^e, f_i^τ, or f_i^δ comprises a linear layer and rectified linear unit. The edge function in the first GNN block, f_1^e, also contains a residual network bottleneck (He et al., 2016), where the convolution layers have been replaced by linear layers. The functions g_i^τ and g_i^δ are multilayer perceptrons ending with the logistic sigmoid. Their purpose is to act as *gates* for information transfer between different nodes in the graph, similar

to the gates used in the LSTM (Hochreiter & Schmidhuber, 1997). $[\cdot,\cdot]$ denotes concatenation.

The aforementioned formulation has the structure of a Graph Neural Network (GNN) block (Battaglia et al., 2018), with both τ_m and δ_n as nodes, and e_{mn} as edges. These blocks permit information exchange between the embeddings. The blocks deviate slightly from the literature. First, we have two types of nodes and use two different updates for them. This is similar to the work of Brasóand Leal-Taixé (2020) where message passing forward and backward in time uses two different neural networks. Second, the accumulation in the nodes in (1b) and (1c) uses an additional gate, permitting the nodes to dynamically select from which message information should be accumulated. This is sensible in our setting, as for instance class information should be passed from detection to track if and only if the track and detection match well.

We construct our graph neural network by stacking GNN blocks. For added expressivity at small computational cost, we interleave them with blocks where there is no information exchange between different graph elements,

$$e_{mn}^{i+1} = f_i^e(e_{mn}^i) \ , \quad \tau_m^{i+1} = f_i^\tau(\tau_m^i) \ , \quad \delta_m^{i+1} = f_i^\delta(\delta_m^i) \ . \tag{2}$$

Each function f_i^e, f_i^τ, and f_i^δ is a residual network bottleneck (He et al., 2016) where the convolutional layers have been replaced by linear layers. The final GNN will provide us with updated edge embeddings which we use for association of detections to tracks, and updated node embeddings which will be used to score tracks and as input to the GNN in the next frame. For an overview of the final GNN, see *GNN Block i* in Table 8.

Remark 1 Note that in the work by Scarselli et al. (2008), in which GNNs were first proposed, a single GNN block is iteratively applied. We instead use different GNN blocks in each iteration. The reason is twofold. First, the initial detection and edge embeddings are 30-dimensional (assuming 25 object categories and a background class) and two-dimensional, respectively. Different dimensionalities may be desired within the GNN. This is not an issue if the first GNN block is different from the subsequent blocks. Second, using different GNN blocks adds some expressivity to the GNN without requiring additional computations.

3.1.4 Association Prediction

We predict the probability that the track m matches the detection n by feeding the edge embeddings e_{mn} through a logistic model

$$\Pr(m \text{ matches } n) = \text{sigmoid}(w \cdot e_{mn} + b) \ . \tag{3}$$

If the probability is high, they are considered to match and the track will obtain the segmentation of that detection. Tracks that do not match any detection with a sufficiently high probability are deemed to have disappeared. These tracks are marked as *inactive*. New tracks are initialized in a similar fashion. The edge embeddings e_{0n} are fed through another logistic model to predict the probability that the detection n should initialize a new track. If the probability is beyond a threshold, a new track is initialized with the embedding of that detection δ_n. This threshold is intentionally selected to be quite low. This leads to additional false positives, but our model can mark them as such by giving them low class scores and not assigning any segmentation pixels to them (see Appendix 1).

Note that we treat the track-detection association as multiple binary classification problems. This may lead to a single detection being assigned to multiple tracks. An alternative would be to instead consider the classification of a single detection as a multiclass classification problem. We observed, however, that this led to slightly inferior results and that it was uncommon for a single detection to be assigned to more than one track.

3.2 Modelling Appearance

In order to accurately match tracks and detections, we create instance-specific appearance models for each tracked object. To this end, a small neural network is applied to the feature maps produced by the instance segmentation backbone. More precisely, the last two outputs of the ResNet (He et al., 2016) backbone, usually referred to as `conv_4x` and `conv_5x`, are processed by a single convolutional layer each. The resulting feature maps are then concatenated and mask-pooled with the masks provided by the instance segmentation method. This yields a single feature vector for each detection. These feature vectors are then processed by a single residual network bottleneck (see *appearance network* in Table 8). The output, x, describes the appearance of the detection. The tracks gather appearance descriptors from the detections and over time construct an appearance model. The similarity in appearance between a track and a detection will serve as an important additional cue during matching. The aim for the appearance network is to learn a rich representation that allows the model to discriminate between visually or semantically similar instances.

Our initial experiments of integrating appearance information directly into the GNN, similar to previous work (Brasó & Leal-Taixé, 2020; Sarlin et al., 2020; Weng et al., 2020), did not lead to noticeable improvement. This is likely due to differences between the problems. The video instance segmentation problem is fairly unconstrained, *i.e.*, there is significant variation in scenes and objects considered. At the same time, video instance segmentation benchmarks con-

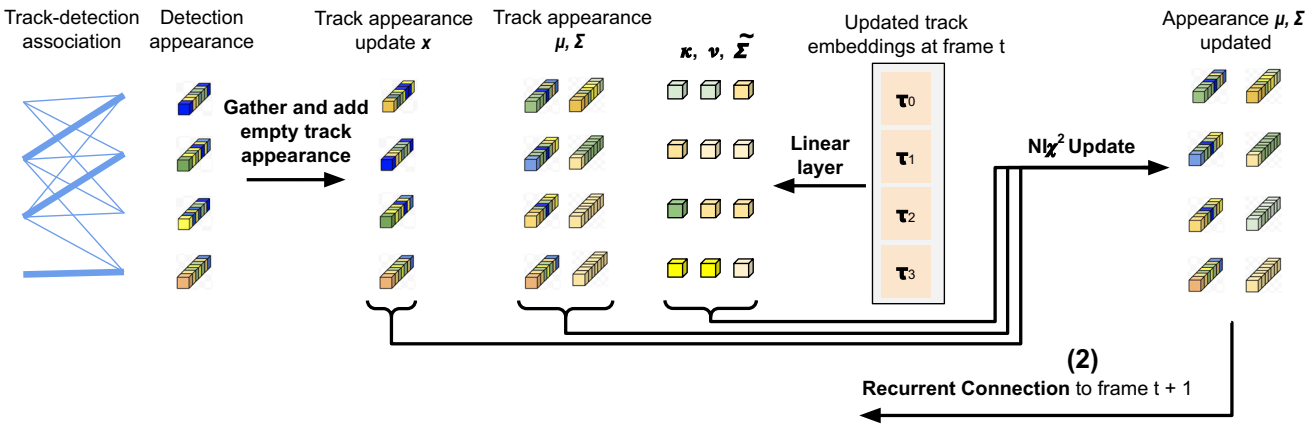

Fig. 4 Illustration of the appearance update process. Each track gathers appearance from its associated detection, as predicted by the GNN. The empty track obtains an appearance corresponding to the entire scene. Each track updates their appearance model with the new appearance update vector following the Bayesian update of a Gaussian under a suitable prior

tain a fairly low number of labelled training sequences. In contrast, multiple object tracking typically works with a single type of scene or a single category of objects. Feature matching, on the other hand, is learnt with magnitudes more training examples than what is available for video instance segmentation.

In order to sidestep this issue, we treat appearance separately and allow the GNN to observe *only* the appearance similarity, and *not* the actual appearance. That is, appearance information is *not* included in the embeddings $\{\delta_n\}_n$ or $\{\tau_m\}_m$. To this end, each track models its appearance using a simple probabilistic model. The GNN observes only the log-likelihood of a detection appearance vector, x, given a track appearance model. This is realized during the construction of the graph, where each edge is initialized with this log-likelihood, as shown in Fig. 2.

We adopt the multidimensional Gaussian distribution with diagonal covariance as track appearance model. When the track is initialized, we take the appearance vector of the initializing detection as mean μ. The entries in the diagonal covariance matrix Σ is initialized with a single value σ^2, which is a learnable parameter of the model. In each frame, the appearance (μ, Σ) of each track is updated with the appearance x of the best matching detection. The appearance models of inactive tracks are not updated. Also the empty track maintains an appearance model. The appearance x used to update this model is obtained by replacing the mask-pooling with average-pooling.

The appearance update is based on the Bayesian update of a Gaussian under a conjugate prior. We use a normal-inverse-chi-square prior (NIχ^2) (Murphy, 2007),

$$\mu^+ = \kappa x + (1-\kappa)\mu \ , \tag{4a}$$

$$\Sigma^+ = \nu\tilde{\Sigma} + (1-\nu)\Sigma + \frac{\kappa(1-\nu)}{\kappa+\nu}(x-\mu)^2 \ . \tag{4b}$$

The term $\tilde{\Sigma}$ corresponds to the sample variance and the update rates κ and ν would usually be the number of samples in the update relative the strength of the prior. For added flexibility we predict these values by applying a linear layer to each track embedding, τ_m, permitting the network to learn a good update strategy. For the sample variance, $\tilde{\Sigma} = \tilde{\sigma}^2 I$, the model predicts a single value that is broadcast along the feature dimension. The appearance update process is illustrated in Fig. 4.

3.3 Recurrent Connection

In order to process object tracks, it is crucial to propagate information over time. We achieve this with a recurrent connection, which brings the benefit of end-to-end training. However, naïvely adding recurrent connections leads to highly unstable training and in extension, poor video instance segmentation results (see Table 3). Even with careful weight initialization and low learning rate, both activation and gradient spikes arise. This is a well-known problem when training recurrent neural networks and is usually tackled with the Long Short-Term Memory (LSTM) (Hochreiter & Schmidhuber, 1997) or Gated Recurrent Unit (Cho et al., 2014). These modules use a system of multiplicative sigmoid-activated gates, and have been repeatedly shown to be able to well model sequential data while avoiding aforementioned issues (Hochreiter & Schmidhuber, 1997; Cho et al., 2014; Greff et al., 2016).

We adapt the LSTM to our scenario. Typically, the output of the LSTM is fed as its input in the next time step. We instead feed the output of the LSTM as input to the GNN in the next time step, and the output of the GNN as input to the LSTM. First, with abuse of notation, denote the output of the GNN as

$$\{\tilde{\tau}_m^t\}, \{\tilde{\delta}_n^t\}, \{\tilde{e}_{mn}^t\} = \text{GNN}(\{\tau_m^{t-1}\}, \{\delta_n^t\}, \{e_{mn}^t\}) \ , \tag{5}$$

where superscript t denotes time. The sets contain elements for all indices $n = 1, \ldots, N$ and $m = 0, \ldots, M$. Next, we feed each track embedding $\tilde{\tau}_m^t$ through the LSTM system of gates

$$\alpha_m^{\text{forget}} = \sigma(h^{\text{forget}}(\tilde{\tau}_m^t)) \ , \tag{6a}$$
$$\alpha_m^{\text{input}} = \sigma(h^{\text{input}}(\tilde{\tau}_m^t)) \ , \tag{6b}$$
$$\alpha_m^{\text{output}} = \sigma(h^{\text{output}}(\tilde{\tau}_m^t)) \ , \tag{6c}$$
$$\tilde{c}_m^t = \tanh(h^{\text{cell}}(\tilde{\tau}_m^t)) \ , \tag{6d}$$
$$c_m^t = \alpha_m^{\text{forget}} \odot c_m^{t-1} + \alpha_m^{\text{input}} \odot \tilde{c}_m^t \ , \tag{6e}$$
$$\tau_m^t = \alpha_m^{\text{output}} \odot \tanh(c_m^t) \ . \tag{6f}$$

The functions $h^{\text{forget}}, h^{\text{input}}, h^{\text{output}}, h^{\text{cell}}$ are linear neural network layers. $\odot$ is the element-wise product, tanh the hyperbolic tangent, and σ the logistic sigmoid. Note that the recurrent module is recurrent over time. In each frame, the system of gates (6) is applied once.

3.4 VIS Output Prediction

3.4.1 Track Scoring

For the VIS task, we need to constantly assess the validity and class membership of each active track τ_m. To this end, we predict a confidence value and the class of existing tracks in each frame. The confidence reflects our trust about whether or not the track is a true positive. It is updated over time together with the class prediction as more information becomes available. This provides the model with the option of effectively removing tracks by reducing their scores. Existing approaches (Cao et al., 2020; Yang et al., 2019; Luiten et al., 2019; Bertasius & Torresani, 2020) score tracks by averaging the detection confidence of the detections deemed to correspond to the track. Class predictions are made with a majority vote. The drawback is that other available information, such as how certain we are that each detection indeed belongs to the track or the consistency of the detections, is not taken into account.

We address the problem of track scoring and classification using the GNN introduced in Sect. 3.1 together with a recurrent connection (Sect. 3.3). The track embeddings $\{\tau_m\}_{m=1}^M$ gather information from all detections via the GNN, and accumulate this information over time via the recurrent connection. We then predict the confidence and class for each track based on its embedding. This is achieved via linear layer followed by softmax.

Remark 2 The proposed approach is *causal* or *online*. In other words, predictions made for frame L are based solely on past frames $l \leq L$. When a prediction is made for a frame L, no future frames $l > L$ are utilized. However, video instance segmentation performance is usually measured in terms of VIS-AP (Yang et al., 2019; Qi et al., 2021), which relies on only a single class and confidence prediction for each track. Prior works typically report segmentations in an online fashion but break causality by averaging confidences or majority voting on the class over all frames (Yang et al., 2019; Cao et al., 2020; Yang et al., 2021). For VIS-AP computation, these methods are not strictly online, but these methods would function in an online setting. We therefore still refer to them as online. This follows the nomenclature used in, *e.g.*, Yang et al. (2019) or Yang et al. (2021).

3.4.2 Segmentation

In each frame, we report a segmentation in which each pixel contains the index of a track or zero (corresponding to background) (Fig. 5). We construct this segmentation from the associated detections—with accompanying masks—and the track embeddings. The latter enable the model to prioritize among tracks when multiple associated masks overlap. We found this critical in scenarios where the model has initialized multiple tracks for a single object.

For each track, we gather the box—represented as a mask—and raw mask scores of the associated detection. The raw mask scores are the mask logits predicted by the detector. We feed the track embedding through a linear layer and a rectified linear unit to reduce the number of channels and then spatially broadcast the result to the mask height and width. Finally, the box masks, raw mask scores, and the broadcast track embeddings are stacked and fed through two convolutional layers to produce a reweighted segmentation score for each track. We call this the *mask reweighting module*. The final segmentation is constructed by applying the `argmax` operator to the reweighted segmentation scores over all tracks. If no track has a reweighted segmentation score greater than zero for a given pixel, that pixel is set as background.

3.5 Training

We train the neural network to initialize new tracks and make predictions for existing tracks. This is achieved by feeding a sequence of T frames through the neural network as we would during inference at test-time. In each frame t, the neural network predicts track-detection match probabilities, track initialization probabilities, track class probabilities, and track segmentation probabilities

$$\mathbf{y}_t^{\text{match}} \in [0, 1]^{M_t \times N_t} \ , \tag{7a}$$
$$\mathbf{y}_t^{\text{init}} \in [0, 1]^{N_t} \ , \tag{7b}$$

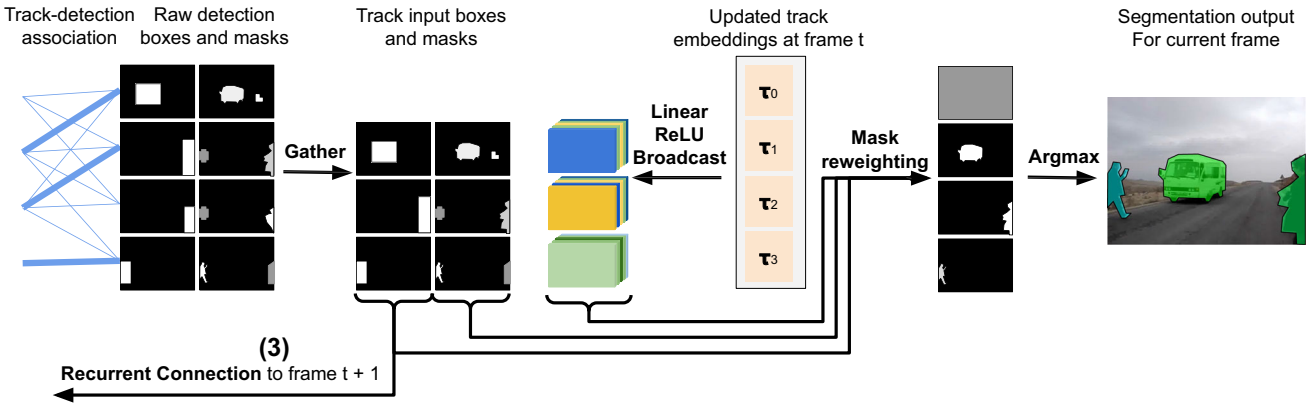

Fig. 5 Illustration of the segmentation prediction. Each track gathers the box and mask from its associated detection. These are then concatenated with the broadcast track embedding and fed through two convolutional layers, providing new segmentation scores

$$\mathbf{y}_t^{\text{score}} \in [0,1]^{M_{t+1} \times C} \ , \tag{7c}$$

$$\mathbf{y}_t^{\text{seg}} \in [0,1]^{M_{t+1} \times H \times W} \ . \tag{7d}$$

Here, M_t denotes the number of tracks in frame t prior to initializing new tracks; N_t the number of detections obtained from the detector in frame t; C the number of object categories, including background; and $H \times W$ the image size. The four components in (7) permit the model to conduct video instance segmentation. We penalize each with a corresponding loss component

$$\mathcal{L} = \lambda^1 \mathcal{L}^{\text{score}} + \lambda^2 \mathcal{L}^{\text{seg}} + \lambda^3 \mathcal{L}^{\text{match}} + \lambda^4 \mathcal{L}^{\text{init}} \ . \tag{8}$$

The component $\mathcal{L}^{\text{score}}$ rewards the network for correct prediction of the class scores; $\mathcal{L}^{\text{seg}}$ for segmentation refinement; $\mathcal{L}^{\text{match}}$ for assignment of detections to tracks; and $\mathcal{L}^{\text{init}}$ for initialization of new tracks. We weight the components with constants $(\lambda^1, \lambda^2, \lambda^3, \lambda^4)$.

In order to compute the loss, we determine the identity of each track and each detection. The identity is either one of the annotated objects or background. First, for each frame, the detections are matched to the annotated objects in that frame. Detections can claim the identity of an annotated object if their bounding boxes overlap by at least 50%. If multiple detections overlap with the same object, only the best matching detection claims its identity. Detections that do not claim the identity of an annotated object are marked as background. Thus, each annotated object will correspond to a maximum of one detection in each frame. Next, the tracks are assigned identities. Each track was initialized by a single detection at some frame and the track can claim the identity of that detection. However, if multiple tracks try to claim the identity of a single annotated object, only the first initialized of those tracks gets that identity. The others are assigned as background. Thus, each annotated object will correspond to a maximum of one track. This procedure is illustrated in Fig. 6.

Using the track and detection identities we compute the loss components. Each component is normalized with the batchsize and video length, but not with the number of tracks or detections. Detections or tracks that are false positives will therefore not reduce the loss for other tracks or detections, as they otherwise would.

$\mathcal{L}^{\text{match}}$ is the binary cross-entropy loss. The target for $\mathbf{y}_{t,m,n}^{\text{match}}$ is 1 if track m and detection n has the same identity and that identity corresponds to an annotated object. If their identities differ or if the identity is background, the target is 0.

$\mathcal{L}^{\text{init}}$ is the binary cross-entropy loss. The target for $\mathbf{y}_{t,n}^{\text{init}}$ is 1 if detection n initializes a track with the identity of an annotated object. Otherwise, the target is 0.

$\mathcal{L}^{\text{score}}$ is the cross-entropy loss. If track m corresponds to an annotated object, the target for $\mathbf{y}_{t,m}^{\text{score}}$ is the category of that object. Otherwise the target is the background class. We found that it was difficult to score tracks early on in some scenarios and therefore we weight the loss over the sequence, giving higher weight to later frames. We assign a weight of 0.8^{L-l} for the lth frame and then normalize the weights such that they sum to one.

$\mathcal{L}^{\text{seg}}$ is the Lovasz loss (Berman et al., 2018). The target for $\mathbf{y}_t^{\text{seg}}$ is obtained by mapping the annotated object identities in the ground-truth segmentation to the track identities. In scenarios where a single annotated object gives rise to multiple tracks, the network is rewarded for assigning pixels only to the track that claimed the identity of that object.

3.6 Generalizing to Longer Sequences and Crowded Scenes

A major challenge in video instance segmentation is occlusion. Qi et al. (2021) identified this challenge and proposed to target long-time videos and heavily crowded scenes. The crowded scenes lead to several similar objects close to each other. Throughout the sequence, objects may also appear or

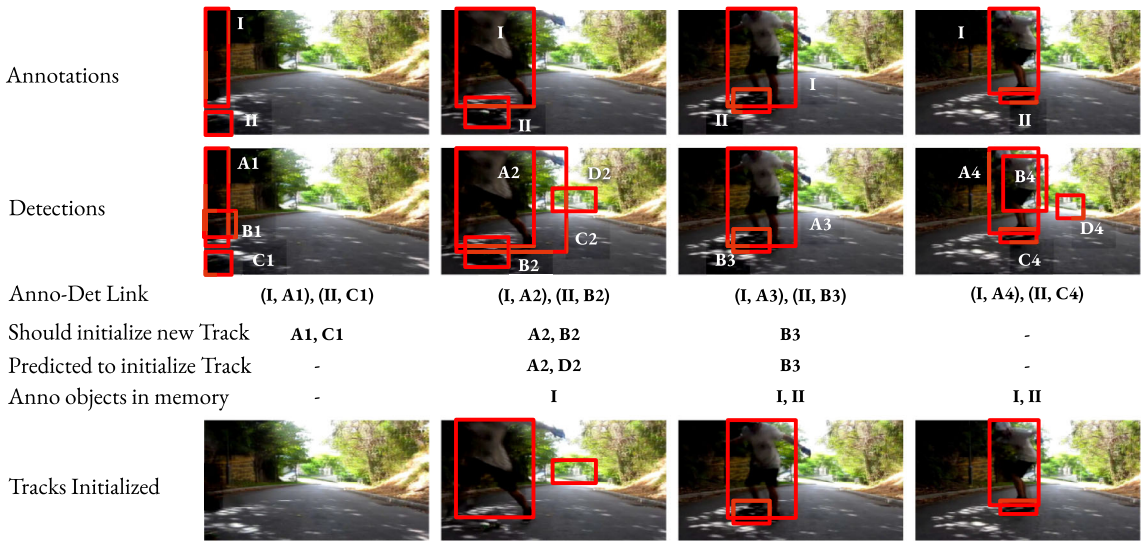

Fig. 6 An illustration of the training procedure on a single sequence of four frames. First, annotated objects are linked to detections in each frame. The model is then applied to the sequence. In each frame, the model predicts whether a detection should be the start of a new track, *i.e.*, if the detection (i) is linked to an annotation and (ii) is not yet added to the track memory. If the prediction is a true positive, the track is linked to the annotation. Otherwise, the track is marked as a false positive. This link between tracks and annotations enables computation of $\mathcal{L}^{\text{score}}$, $\mathcal{L}^{\text{seg}}$, and $\mathcal{L}^{\text{match}}$

disappear, either due to occlusions—which was the focus in the work of Bai et al.—or due to being out of view. In addition, the camera is non-stationary and often exhibits fairly non-smooth motion. These three challenges make the VIS-problem highly challenging and methods need to take a wide variety of information into account. We explore three directions to better deal with such scenarios.

3.6.1 Long Sequences

We argue that for video instance segmentation, a flexible model is desired. The neural network proposed in this work is highly flexible. In each frame, all tracks in memory and all detections are processed jointly. Both spatial and appearance-based information is utilized. Furthermore, the model itself decides what information to put into the track embeddings that are recurrently fed to the next timestep. We believe, however, that in order to fit a flexible VIS-model, two key components are needed. First, the VIS-problem needs to be accurately modelled during training. This is in contrast to, for instance, training for some proxy-task, such as mask-propagation, and then rely on heuristics to produce the final VIS-output. Accurate modelling of the VIS-problem ensures that the neural network is optimized for video instance segmentation. The learning formulation proposed in Johnander et al. (2021) is intended to serve this purpose. Second, we need data that well represents the challenges in VIS. Two such challenges are complex motion and objects that appear or disappear. Typically, these challenges are present mostly in sequences that span over sufficiently long time frames. We therefore experiment with longer sequences during training.

3.6.2 Normalization

In highly crowded scenes, the number of objects can rapidly increase or decrease over time. Moreover, the number of objects in a video depends on whether the scene is crowded or not. This is potentially problematic for the proposed approach. In the proposed approach, the strength of the activations in (1) depends on the number of objects. When there are many objects in the scene, the activations become stronger, exhibiting a higher variance. When there are few objects in the scene, the activations reduce in strength. In such scenarios, we hypothesize that the model benefits from normalization layers, in order to stabilize the activations. Therefore, we propose to normalize the activations using layer normalization (LN) Ba et al. (2016). The GNN equations instead become

$$e_{mn}^{i+1} = \text{LN}(f_i^e([e_{mn}^i, \tau_m^i, \delta_n^i])) \ , \tag{9a}$$

$$\tau_m^{i+1} = \text{LN}(f_i^\tau([\tau_m^i, \sum_j g_i^\tau(e_{mj}^i) e_{mj}^i])) \ , \tag{9b}$$

$$\delta_n^{i+1} = \text{LN}(f_i^\delta([\delta_n^i, \sum_j g_i^\delta(e_{jn}^i) e_{jn}^i])) \ . \tag{9c}$$

In layer normalization, it is possible to select the dimensions to normalize over. Here, we normalize over the embedding channel dimension only. Note that this effectively incorpo-

rates normalization over the number of tracks and detections as the GNN layers sum over those dimensions.

3.6.3 Positional Encodings

Last, inspired by the success of transformers for other tasks (Vaswani et al., 2017; Carion et al., 2020), we experiment with positional encodings. Their original purpose was to encode positional or spatial information for the otherwise positional or spatially equivariant transformer. For object detection, it has been shown that the introduction and design of positional encodings plays a crucial role for convergence and performance (Carion et al., 2020; Meng et al., 2021; Zhu et al., 2020). We experiment with the addition of positional encodings, encoding some spatial information into the otherwise spatially invariant appearance vector. We adopt the encodings used in the work of DETR (Carion et al., 2020). During the construction of the appearance vector x, the positional encodings are added prior to the mask-pooling operation.

$$x = \phi(\text{MaskPool}(\psi(I) + p)) \ . \tag{10}$$

Here, I denotes the image and ψ the ResNet backbone used in the detector with an added convolutional layer. The positional encoding $p \in \mathbb{R}^{D \times H \times W}$ is added to the output of ψ before mask pooling with the detection mask. The result is fed through a multilayer perceptron ϕ to produce the final appearance vector.

4 Experiments

We evaluate the proposed approach for video instance segmentation on two benchmarks. The first one, YouTube-VIS (Yang et al., 2019), is a benchmark comprising 40 object categories in 2k training videos and 300 validation videos. The second one, OVIS (Qi et al., 2021), is a more challenging benchmark comprising 25 object categories in 607 training videos and 150 validation videos. The performance is measured by both benchmarks in terms of video mean average precision (AP). We first provide qualitative results, showing that the proposed neural network learns to tackle the video instance segmentation problem for both benchmarks. Next, we quantitatively compare to the state-of-the-art. Then, we analyze the different components and aspects of our approach in an ablation study. Last, we conduct an analysis on long and highly crowded sequences.

4.1 Implementation Details

We implement the proposed approach in PyTorch (Paszke et al., 2017). We aim for real-time performance and therefore select YOLACT (Bolya et al., 2019) as base instance segmentation method. We use the implementation publicly provided by the authors. We adopt a ResNet50 or ResNet101 backbone (He et al., 2016). The detector and backbone are initialized with weights provided with the YOLACT implementation. We fine-tune the detector on images from the YouTubeVIS and OpenImages (Kuznetsova et al., 2020; Benenson et al., 2019) training sets for 120 epochs à 933 iterations, with a batch size of 8. Next, we freeze the backbone and the detector, and train all other modules: the appearance network, the GNN, and the recurrent module. We train for 150 epochs à 633 iterations with a batch of 4 video clips (10 frames each) sampled randomly from YouTubeVIS. During training, 200 sequences of YouTubeVIS are held-out for hyperparameter selection. When generalizing to longer sequences and crowded scenes, we instead fine-tune the detector on images from OVIS and OpenImages (Kuznetsova et al., 2020; Benenson et al., 2019) training sets. We train equally many iterations with a batch size of 8. All other modules train for 300 epochs with a batch of 4 video clips. Each consists of 20 frames sampled randomly from OVIS.

During training we use dropout regularization on all feature maps extracted by the backbone, dropping channels with probability 0.1, before feeding them into the instance segmentation method and our appearance network. We adopt two GNN blocks, corresponding to (1) or (9), each followed by a residual block, corresponding to (2). When the graph is constructed, the edge embeddings e_{mn} are 2-dimensional. The detection embeddings δ_n are 45-dimensional for the YouTubeVIS experiments and 30-dimensional for the OVIS experiments. After the first GNN-block has been applied, all embeddings are 128-dimensional. The track embeddings τ_m are produced by the model in the previous frame and are thus always 128-dimensional. This dimensionality is unchanged in the recurrent module. The appearance network reduces the dimensionality of `conv4_x` and `conv5_x` to 256 channels each, before being pooled and concatenated. The resulting appearance descriptor is 512-dimensional. All residual network bottlenecks reduce the channels by a factor of 4 within the bottleneck. In the mask reweighting module, used to predict the segmentation, the track embeddings are projected down to 16 dimensions before being spatially broadcasted.

We set the track initialization threshold probability to 0.13 (`softplus(−2)`) during training and 0.31 (`softplus(−1)`) during inference. The low training threshold leads to many false positives, especially early on in training. During our early experiments, we found this to produce a better model. During inference, entire sequences are processed rather than video clips. In order to protect the track memory from becoming full, a slightly higher threshold is selected. Tracks are marked as inactive if there is no detection that matches with a probability of at least 0.31. For the post-processing inside YOLACT, we keep detections with a

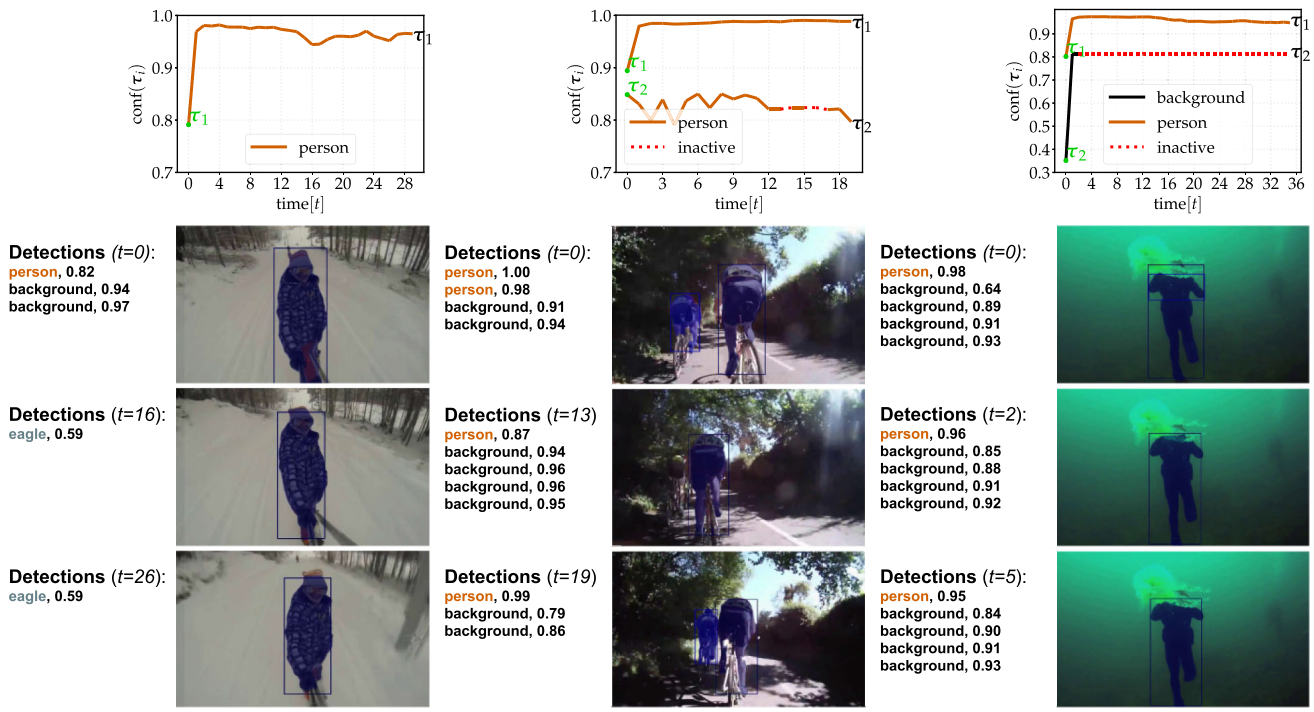

Fig. 7 Track score plots (top) and detections (3 bottom rows) for three videos. The plot colour is the ground-truth class for that track and its confidence is shown on the y-axis, ideally 1.00. In the left video, the detector makes noisy class predictions, but our approach learns to filter this noise. In the center, there is a missed detection. Our method renders the track inactive and resumes it in subsequent frames where the detector finds both objects. To the right, a false positive in the detector leads to a false track. This track is, however, quickly marked as background with high confidence

confidence of at least 0.03 for any class, and set the non-maximum suppression intersection over union threshold to 0.7. We found that these values led to a good performance of our final method on the 200 videos we held out for validation.

4.2 Qualitative Results

In Fig. 7 we show the output of the detector and the tracks predicted by our approach on the YouTubeVIS validation dataset (Yang et al., 2019). The detector may provide noisy class predictions. Our model learns to filter these predictions and accurately predicts the correct class. When the detector fails to detect an object, our approach pauses the corresponding track until the detector finds the object again. If the detector provides a false positive, our approach initializes a track that is later marked as background and rendered inactive. The proposed model has learnt to deal with mistakes made by the detector. For additional qualitative results, see the Appendix.

4.3 Quantitative Comparison

Next, we compare our approaches RGNNVIS and RGNNVIS++ to the state-of-the-art, including the baselines proposed in Yang et al. (2019). The results are shown in Tables 1 and 2. Our approach, RGNNVIS, running at 30 fps, outperforms all near real-time methods. DeepSORT (Wojke et al., 2017), which relies on Kalman-filtering the bounding boxes and a learnt appearance descriptor used for re-identification, obtains an AP score of 26.1. MaskTrack R-CNN (Yang et al., 2019) gets a score of 30.3. SipMask (Cao et al., 2020) improves MaskTrack R-CNN by changing its detector and reaching a score of 33.7. Using a ResNet50 backbone, we run at a similar speed and outperform all three methods with an absolute gain of 9.2, 5.0, and 1.6 AP, respectively.

While some methods (Luiten et al., 2019; Bertasius & Torresani, 2020) obtain higher AP, those methods are more than a magnitude slower, and thus infeasible for real-time applications or for processing large amounts of data. STEm-Seg (Athar et al., 2020) reports results using both a ResNet50 and a ResNet101 backbone. We show a gain of 4.7 AP with ResNet50. We also try with a ResNet101 backbone, retraining our base detector and approach. This leads to a performance of 37.7 AP, an absolute gain of 3.1 AP. RGNNVIS++ extends RGNNVIS, adopting the extensions proposed in Sect. 3.6. Our RGNNVIS++ method did not lead to better performance on YouTubeVIS, as shown in Table 6

Table 1 State-of-the-art comparison on the YouTubeVIS validation dataset (Yang et al., 2019)

Method	Backbone	Online	FPS	AP
FEELVOS (Voigtlaender et al., 2019)	ResNet50			26.9
SeqTracker (Yang et al., 2019)	ResNet50			27.5
STEm-Seg (Athar et al., 2020)	ResNet50	Near		30.6
OSMN MaskProp (Yang et al., 2018)	ResNet50	✓		23.4
IoUTracker+ (Yang et al., 2019)	ResNet50	✓		23.6
OSMN (Yang et al., 2018)	ResNet50	✓		27.5
DeepSORT (Wojke et al., 2017)	ResNet50	✓		26.1
MaskTrack R-CNN (Yang et al., 2019)	ResNet50	✓	20	30.3
SipMask (Cao et al., 2020)	ResNet50	✓	30	32.5
SipMask ms-train (Cao et al., 2020)	ResNet50	✓	30	33.7
RGNNVIS (Ours)	ResNet50	✓	30	35.3
RGNNVIS++ (Ours)	ResNet50	✓	30	35.1
VIS2019 Winner (Luiten et al., 2019)	ResNext101-32x48d		$< 1^{\dagger}$	44.8
MaskProp (Bertasius & Torresani, 2020)	ResNext101-64x4d		$< 2^{\dagger}$	46.6
STEm-Seg (Athar et al., 2020)	ResNet101	Near	7	34.6
RGNNVIS (Ours)	ResNet101	✓	25	37.7
RGNNVIS++ (Ours)	ResNet101	✓	25	37.7

The proposed approach outperforms all near real-time approaches. †: No speed reported in Luiten et al. (2019) or Bertasius and Torresani (2020), but each utilize components (Luiten et al. 2020 and Chen et al. 2019) with a reported speed of 1 fps and 2 fps respectively

Table 2 Extension of Table 1 on the YouTubeVIS validation datasets (Yang et al., 2019) for recently proposed approaches

Method	Backbone	Online	AP	AP_{50}	AP_{75}	AR_1	AR_{10}
CMaskTrack R-CNN (Qi et al., 2021)	ResNet50		32.1	52.8	34.9	33.2	37.9
CSipMask (Yang et al., 2019)	ResNet50		35.1	55.6	38.1	35.8	41.7
VisTR (Wang et al., 2021)	ResNet50		35.6	56.8	37.0	35.2	40.2
VisIFCT (Hwang et al., 2021)	ResNet50		41.2	65.1	44.6	42.3	49.6
STMask (ada) (Li et al., 2021)	ResNet50	✓	33.5	52.1	36.9	31.1	39.2
CrossVIS (Yang et al., 2021)	ResNet50	✓	36.3	56.8	38.9	35.6	40.7
RGNNVIS (Ours)	ResNet50	✓	35.3	53.9	39.1	32.6	38.1
RGNNVIS++ (Ours)	ResNet50	✓	35.1	52.9	39.9	31.8	39.0
VisTR (Wang et al., 2021)	ResNet101		38.6	61.3	42.3	37.6	44.2
VisIFCT (Hwang et al., 2021)	ResNet101		42.6	66.6	46.3	43.5	51.4
CrossVIS (Yang et al., 2021)	ResNet101	✓	36.6	57.3	39.7	36.0	42.0
STMask (ada) (Li et al., 2021)	ResNet101	✓	36.8	56.8	38.0	34.8	41.8
RGNNVIS (Ours)	ResNet101	✓	37.7	57.5	41.8	34.5	41.4
RGNNVIS++ (Ours)	ResNet101	✓	37.7	58.6	41.9	34.7	41.7

4.4 Ablation Study

In this section we analyze the different aspects of the proposed approach on the YouTubeVIS validation dataset (Yang et al., 2019), with results provided in Table 3.

4.4.1 No GNN

We first analyze the benefit of processing tracks and detections jointly using our GNN. This is done by restricting the GNN module. First, a neural network predicts the probability that each track-detection pair matches, based only on the appearance and spatial similarities. Next, new tracks are initialized from detections that are not assigned to any track. Last, each track embedding is updated with the best matching edge and detection. This leads to a substantial drop in AP (6.7% absolute), demonstrating the importance of our GNN.

4.4.2 MLP Node Updates

In our approach we utilize sigmoid gates in the node updates. As mentioned in Sect. 3.1, the purpose of these gates is to

Table 3 Performance under different configurations on the YouTubeVIS validation set

Configuration	AP
Our final approach	35.3
No GNN	28.6
MLP Node Updates	34.1
1 GNN Block	32.4
3 GNN Blocks	35.2
No Interleaved Residual Blocks	33.3
No LSTM-like gating	Diverges
Simple recurrent gate	31.5
No appearance	34.5
Appearance baked into embedding	34.4
Appearance const. variance	33.0
Appearance with LSTM	34.0
Association from Yang et al. (2019); Cao et al. (2020)	29.2
Scoring from Yang et al. (2019); Cao et al. (2020)	31.5
Scoring as average	30.7

Each experiment corresponds to a single alteration to the final approach. The first set of experiments seeks to simplify the different modules in the final approach. The second set of experiments tackles the association and scoring tasks of the VIS-problem using the mechanism proposed by Yang et al. (2019); Cao et al. (2020)

permit a node to dynamically select from which other nodes it wants to gather information. This is in contrast to for instance (Brasó & Leal-Taixé, 2020) where a 2-layer multilayer perceptron (MLP) is utilized. We experiment with removing the gates and instead adding an extra layer to f^e, f^τ, and f^δ, making them into 2-layer MLPs. In doing so, we observe a drop of 1.2 mAP.

4.4.3 Number of GNN Blocks

As mentioned in Sect. 4.1, we opted to use two GNN blocks where information is passed between different graph elements. This is the same as the length of the longest path in our bipartite graph, and we are therefore able to feed information from any graph element to any other graph element. We try to instead use a single GNN block and to use three GNN blocks. With a single GNN block, and thus with a model unable to propagate information between any pair of graph elements, we observe a drop of 2.9 mAP. Using three GNN blocks, compared to using two GNN blocks, leads to a minor drop of 0.1 mAP.

4.4.4 No Interleaved Residual Blocks

As argued in Sect. 3.1, we interleave the GNN blocks with residual blocks in order to add some flexibility to the GNN. We try to run without them. This leads to a drop of 2.0 mAP.

4.4.5 Simpler Recurrent Module

We experiment with the LSTM-like gating mechanism. We first try to remove it, directly feeding the track embeddings output from the GNN as input in the subsequent frame. We found that this configuration leads to unstable training and in all attempts diverge. We therefore also try a simpler mechanism, adding only a single gate and a tanh activation. This setting leads to more stable training, but provides deteriorated performance.

4.4.6 Simpler Appearance

We measure the impact of the appearance by removing it. We also experiment with removing its separate treatment. The appearance is instead baked into the detection node embeddings. Both of these configurations lead to performance drops. We also try to replace the covariance estimates with a constant variance. This leads to a 2.3 drop in mAP. Moreover, we feed the appearance vectors into an LSTM that directly predicts the mean and covariance vectors. This leads to a 1.3 drop in mAP.

4.4.7 Association or Scoring from Yang et al. (2019)

The proposed model is trained to (i) associate detections to tracks and (ii) score tracks. We try to let each of these two tasks instead be performed by the simpler mechanisms used in previous works (Yang et al., 2019; Cao et al., 2020). This leads to performance drops of 6.1 and 3.8 AP respectively. Our YOLACT detector also provides class and confidence

Table 4 Results of state-of-the-art methods on the OVIS validation dataset (Qi et al., 2021)

Method	Backbone	Online	AP	AP_{50}	AP_{75}	AR_1	AR_{10}	AP_{SO}	AP_{MO}	AP_{HO}
CSipMask (Yang et al., 2019)	ResNet50		14.3	29.9	12.5	9.6	19.3	27.1	16.6	3.2
CMaskTrack R-CNN (Qi et al., 2021)	ResNet50		15.4	33.9	13.1	9.3	20.0	28.6	18.7	4.1
STEM-Seg (Athar et al., 2020)	ResNet50	Near	13.8	32.1	11.9	9.1	20.0	22.2	16.1	3.9
FEELVOS (Voigtlaender et al., 2019)	ResNet50	✓	9.6	22.0	7.3	7.4	14.8	17.3	11.5	1.7
IoUTracker+ (Yang et al., 2019)	ResNet50	✓	7.0	16.9	5.3	5.7	14.3	11.5	7.9	1.8
SipMask (Cao et al., 2020)	ResNet50	✓	10.2	24.7	7.8	7.9	15.8	19.9	10.5	2.2
TraDeS (Wu et al., 2021)	ResNet50	✓	11.4	26.5	9.4	7.0	13.8	23.0	12.8	3.0
QueryInst-VIS (Fang et al., 2021)	ResNet50	✓	14.7	34.7	11.6	9.0	21.2	27.3	17.2	4.1
CrossVIS (Yang et al., 2021)	ResNet50	✓	14.9	32.7	12.1	10.3	19.8	28.4	16.9	4.1
MaskTrack R-CNN (Yang et al., 2019)	ResNet50	✓	10.8	25.3	8.5	7.9	14.9	23.0	12.8	2.7
STMask (Li et al., 2021)	ResNet50	✓	15.4	33.8	12.5	8.9	21.3	24.0	18.7	5.1
RGNNVIS (Ours)	ResNet50	✓	13.8	28.0	12.6	8.6	18.7	25.0	15.4	4.3
RGNNVIS++ (Ours)	ResNet50	✓	16.0	33.3	13.1	9.2	21.2	26.4	17.8	4.8
RGNNVIS (Ours)	ResNet101	✓	14.2	28.6	12.0	8.3	19.5	25.9	17.0	4.1
RGNNVIS++ (Ours)	ResNet101	✓	16.3	32.8	14.6	10.3	21.7	26.5	19.4	4.6

We extend our method to better deal with scenarios of occlusions and crowded scenes. Our model then outperforms all previous approaches. The result for light, medium, and high occlusion levels are reported as AP_{SO}, AP_{MO}, AP_{HO}

Table 5 Performance of the proposed approach (using a ResNet50 backbone) on the OVIS validation set, exhibiting long videos and crowded scenes

Long sequences	Normalization	Positional encodings	AP	AP_{50}	AP_{75}	AR_1	AR_{10}	AP_{SO}	AP_{MO}	AP_{HO}
			13.8	28.0	12.6	8.6	18.7	25.0	15.4	4.3
✓			13.5	28.0	11.1	8.5	18.9	23.4	14.7	4.1
	✓		14.1	27.0	12.9	8.0	19.7	23.2	15.6	4.2
✓	✓		15.8	32.1	13.9	9.4	20.6	26.7	18.0	5.0
✓	✓	✓	16.0	33.3	13.1	9.2	21.2	26.4	17.8	4.8

On this challenging dataset, the proposed approach benefits from training with longer sequences, from normalization inside the GNN, and from positional encodings

Table 6 Performance of the proposed approach (using a ResNet50 backbone) on the YouTubeVIS validation set

Long sequences	Normalization	Positional encodings	AP	AP_{50}	AP_{75}	AR_1	AR_{10}
			35.3	53.9	39.1	32.6	38.1
✓			34.6	53.1	39.4	32.0	38.0
	✓		35.9	54.4	41.1	32.6	39.3
✓	✓		35.7	55.3	40.5	32.4	39.3
✓	✓	✓	35.1	52.9	39.9	31.8	39.0

Using this dataset, training with longer sequences or adding positional encodings did not benefit the proposed method

predictions as a single vector, containing a score for each class and for the background. We try to create the track score by directly averaging these vectors. The class membership and confidence are found as in YOLACT via a `softmax` followed by `argmax` and `max`, respectively. The advantage is that more information of the detector is kept, compared to the mechanism in Yang et al. (2019); Cao et al. (2020). However, this leads to a 4.6 drop in mAP, quite close to the 3.8 drop we observed when using the same mechanism as (Yang et al., 2019; Cao et al., 2020).

4.5 Study on Occluded Video Instance Segmentation

Qi et al. (2021) use YouTubeVIS as a starting point and identify two directions that make video instance segmentation challenging; (i) longer sequences and (ii) more object crowded scenes. To this end, the Occluded Video Instance

Table 7 Performance of the proposed approach on the OVIS and YouTubeVIS validation sets for different sets of training data

Backbone	OVIS training data	YouTubeVIS training data	AP (OVIS)	AP (YouTubeVIS)
ResNet50		✓		35.3
ResNet50	✓		16.0	
ResNet50	✓	✓	14.8	36.9
ResNet101		✓		37.7
ResNet101	✓		16.3	
ResNet101	✓	✓	14.3	38.0

The addition of OVIS training data substantially improves YouTubeVIS results whereas the addition of YouTubeVIS data harms OVIS results

Segmentation (OVIS) dataset is proposed. This dataset contains fewer videos and fewer object categories, where the set of categories is almost a subset of the categories in YouTubeVIS. Instead, the dataset focuses on having longer sequences and more objects in each sequence. Objects often undergo complex movements at the same time as they occlude each other. As shown by Qi *et al.*, even state-of-the-art methods struggle to robustly track the different objects.

4.5.1 State-of-the-Art Comparison

We compare our approach to the state-of-the-art on OVIS. The results are shown in Table 4. The pioneering work for video instance segmentation, MaskTrackRCNN (Yang et al., 2019), obtains 10.8 AP. This is a dramatic drop from its YouTubeVIS performance of 30.3 AP, demonstrating the challenge of the OVIS dataset. Qi et al. (2021) propose a temporal feature alignment mechanism and apply it to SipMask (Cao et al., 2020) and MaskTrack R-CNN (Yang et al., 2019), obtaining a performance of 14.3 and 15.4 respectively. Our approach obtains a performance of 16.0 AP with a ResNet50 backbone and 16.3 AP with a ResNet101 backbone. The approach performs well on both medium and high occlusion levels. Only STMask (Li et al., 2021) obtains better performance on high occlusion levels. However, compared to STMask, our approach achieves substantially higher performance for lower occlusion levels. This demonstrates the effectiveness of the learning framework and the recurrent graph neural network proposed in this work.

4.5.2 Ablation Study

We experiment with the training sequence length, the normalization inside the GNN, and positional encodings. The results are reported in Table 5. First, we retrain our approach on the OVIS dataset and obtain a score of 13.8 AP. Next, we train with longer sequences, using $L = 20$ instead of $L = 10$. This leads to a performance decrease of 0.3 absolute AP. The decrease is consistent across all occlusion levels. Adding the normalization within the GNN, however, increases the results to 15.8 AP. Adding the positional encodings yields another 0.2 AP increase.

We also run the same experiments on YouTubeVIS and report the results in Table 6. In contrast to OVIS, YouTubeVIS exhibits shorter sequences and less crowded scenes. The added normalization still provides an improvement to performance of 0.6 AP, or 0.4 AP if used together with long sequences. However, adding the positional encodings leads to a performance reduction of 0.6 AP. One possible explanation is that YouTubeVIS has several sequences where the camera is moving wildly (see the middle column in Fig. 7). This makes it possible for a target to move substantially between frames, making the positional encodings unreliable.

4.5.3 Training Data Study

We analyze the performance of our approach when training on the combined training sets of OVIS and YouTubeVIS. The results are reported in Table 7. The YouTubeVIS training set primarily contains short sequences with few objects. In addition, many sequences exhibit simple or little motion. The OVIS training dataset, in contrast, exhibits more crowded scenes and more complex motion. We expect the addition of OVIS training data to greatly aid general video instance segmentation performance. OVIS and YouTubeVIS contain 25 and 40 object categories respectively. 23 of these categories are shared between the datasets. We train our approach to detect the joint set of 42 object categories. On YouTubeVIS, this leads to a substantial improvement from 35.3 AP to 36.9 AP. On OVIS, the addition of YouTubeVIS harms performance, leading to a decrease from 16.0 to 14.8. This demonstrates the importance of challenging training sequences.

4.5.4 Qualitative Comparison

Last, we provide qualitative results for highly crowded scenes and long sequences. In Fig. 8, we show results of RGNNVIS (Johnander et al., 2021) and the extended approach for two sequences from the OVIS benchmark. The sequences are highly challenging with several objects that are visually very

Fig. 8 Qualitative comparison on the challenging OVIS benchmark between RGNNVIS (Johnander et al., 2021) (rows 1 and 3) and the extended approach proposed in this work (rows 2 and 4). The shown frames are taken near the end of the respective videos. In rows 1 and 3, the method fails to create tracks for some objects. In row 3, the cyclist's identity is switched when other cyclists enter the frame from the left. The extended approach, in contrast, creates tracks for all the objects and successfully tracks them

Fig. 9 Qualitative exampes on the OVIS benchmark. In rows 1 and 2, new tracks are successfully created and track identities maintained through severe occlusions. In row 3, the method successfully tracks most objects, but some of the cars driving by are assigned multiple tracks. In row 4, the method successfully tracks until the fourth frame, in which the identity of the bikes switch. Rows 5 and 6 show failure modes where our approach fails to cope with high detector noise

similar. RGNNVIS (Johnander et al., 2021) is in many cases too conservative in its track initialization, missing important objects. In addition, it tends to switch the identities of objects when occlusions occur. In contrast, the extended approach successfully initializes tracks and maintains track identities even in challenging scenarios. In Fig. 9, we provide results of the extended approach for six challenging videos. In the first two videos, the proposed approach initializes tracks for the different objects and successfully tracks them through occlusion and complex motion. The third video shows a failure mode for our approach. Some of the cars driving by obtain multiple tracks. In the fourth video, the two persons and their bikes are successfully tracked through severe occlusions. However, in the fourth shown frame, identities of the bikes are switched. Furthermore, in the third shown frame, a track is incorrectly initialized at the background. The last two videos show scenarios where the detector struggles, providing detections that encompass multiple objects. As our approach relies on the provided detections, it fails in these scenarios.

5 Conclusion

We introduced a novel learning formulation together with an intuitive and flexible model for video instance segmentation. The model proceeds frame by frame, uses as input the detections produced by an instance segmentation method, and incrementally forms tracks. It assigns detections to existing tracks, initializes new tracks, and updates class and confidence in existing tracks. We demonstrate via qualitative and quantitative experiments that the model learns to create accurate tracks, and provide an analysis of its various aspects via ablation experiments.

Supplementary information

We supply a video file `rgnnvis_ovis.mp4` with additional qualitative results on the OVIS validation set (Qi et al., 2021). The video shows results produced by our extended approach with a ResNet50 (He et al., 2016) backbone.

We also supply another video `qualitative_results.mp4` with additional qualitative results. This video shows results produced by our final model (Johnander et al., 2021) with a ResNet101 (He et al., 2016) backbone on the YouTube-VIS (Yang et al., 2019) validation set.

The videos depict a variety of scenarios that our approach is able to handle. From single instances with fast motion, camera movements, multiple similar instances, to crowded scenes. In the end we show some failure cases of our approach. For created tracks we show the top three class probabilities at the bottom of the video, in each of the sequences. Tracks deemed not to match any detection are marked as *inactive*.

Supplementary Information The online version contains supplementary material available at https://doi.org/10.1007/s11263-022-01703-8.

Acknowledgements This work was partially supported by the Wallenberg Artificial Intelligence, Autonomous Systems and Software Program (WASP) funded by Knut and Alice Wallenberg Foundation; and the Excellence Center at Linköping-Lund in Information Technology (ELLIT); and the computations were enabled by the Berzelius resource provided by the Knut and Alice Wallenberg Foundation at the National Supercomputer Centre.

Funding Open access funding provided by Linköping University. The authors have no relevant financial or non-financial interests to disclose.

Declarations

Conflict of interest The authors have no competing interests to declare that are relevant to the content of this article.

Appendix A Model Details

We supply additional details on the training of our model, the model architecture, and how we deal with the sparsity of the tracks and detections.

Model Training Stages

While it is theoretically possible to train our model end-to-end in a single stage, we instead opt for a greedy approach and where the model is trained in two stages. We first train the instance segmentation method and then we train all other components. There are two reasons for this: (i) the instance segmentation method does not benefit from replacing single images with videos and doing so only makes training slower; and (ii) training the entire model together, including the instance segmentation method, on a batch of video clips is prohibitively expensive in terms of memory consumption.

We therefore first train the backbone and the instance segmentation method on single images for 120 epochs. In this stage, we let the batch normalization layers in the backbone

Table 8 Pseudo-code for the construction of the different neural networks of our method

Appearance Network	
processes conv4_x	`Conv2d(1024, 256, 1)`
processes conv5_x	`Conv2d(2048, 256, 1)`
	mask-pool and concatenate
	`Residual(512, 128)`
GNN block 1	
f^e	`Linear(175, 128) + ReLU + Residual(128, 32)`
g^τ	`Linear(128, 32) + ReLU + Linear(32, 128) + Sigmoid`
f^τ	`Linear(256, 128) + ReLU`
g^δ	`Linear(128, 32) + ReLU + Linear(32, 128) + Sigmoid`
f^δ	`Linear(173, 128) + ReLU`
GNN block 2	
f^e	`Residual(128, 32)`
f^τ	`Residual(128, 32)`
f^δ	`Residual(128, 32)`
GNN block 3	
f^e	`Linear(384, 128) + ReLU`
g^τ	`Linear(128, 32) + ReLU + Linear(32, 128) + Sigmoid`
f^τ	`Linear(256, 128) + ReLU`
g^δ	`Linear(128, 32) + ReLU + Linear(32, 128) + Sigmoid`
f^δ	`Linear(256, 128) + ReLU`
GNN block 4	
f^e	`Residual(128, 32)`
f^τ	`Residual(128, 32)`
f^δ	`Residual(128, 32)`
Mask reweighting module	
Processes tracks	`Linear(128, 16) + ReLU`
	broadcast to mask size
	concatenate with detection mask and box (as mask)
	`Conv2d(18, 16, 3) + ReLU + Conv2d(16, 1, 3)`

conv4_x and conv5_x denote the last stride 16 and last stride 32 feature maps of the backbone. `Conv2d(`D^{in}, D^{out}, $K \times K$`)` denotes a 2-dimensional convolutional layer with D^{in} channels in, D^{out} channels out, and a kernel size of K. `Linear(`D^{in}, D^{out}`)` denotes a linear layer. `Residual(`D^{in}, $D^{\text{bottleneck}}$`)` denotes the residual network bottleneck (He et al., 2016), but with linear layers instead of convolutions. The bottleneck comprises three linear layers, the first projecting the input down to $D^{\text{bottleneck}}$ channels and the last transforming the input back to D^{in} channels. `Sigmoid` is the logistic function and `ReLU` the rectified linear unit

keep running averages of the batch statistics. We optimize with Adam, using a batch size of 8, a learning rate of $5 \cdot 10^{-5}$ that is decayed by a factor of 5 after epochs 60 and 90, and a weight decay of 10^{-4}.

We then train all other components for 150 epochs: the appearance network, the graph neural network (GNN), the mask reweighting module, the recurrent gating mechanism, the logistic model for track-detection matching, the track logistic model for track initialization, the track scoring multinomial logistic model, and the appearance update predictors. During this second training stage, the backbone and the instance segmentation are frozen, including the batch statistics of the former. We optimize with Adam, using a batchsize of 4, a learning rate of $2 \cdot 10^{-4}$ and a weight decay of 10^{-4}.

The loss weights $(\lambda^1, \lambda^2, \lambda^3, \lambda^4)$ for $\mathcal{L}^{\text{score}}$, $\mathcal{L}^{\text{seg}}$, $\mathcal{L}^{\text{match}}$, and $\mathcal{L}^{\text{init}}$ are set to (1, 1, 4, 1).

Model Architecture

In Table 8, we supply a list of neural network layers used in the appearance network, in the GNN, and in the mask reweighting module.

Dealing with Sparsity

The tracks and detections vary in number between frames in a single sequence, and between training examples in a batch. We use tensors of fixed size together with a mask,

marking which elements are active and which are not. During for instance aggregation in the nodes, the representation is masked with zeros for inactive edges. We set the maximum size for these tensors to 24 tracks and 16 detections. If the memory contains 24 tracks, no additional tracks will be added. For the detections, we take the 16 detections with highest confidence, as predicted by the instance segmentation method. We found these numbers to be sufficiently large for the datasets used in this work. The proposed approach rarely fills up the memory of tracks and the instance segmentation method rarely produces more than 16 detections with high confidence. For datasets with very long sequences or for real-life deployment, one might desire a larger memory or some track manangement system that discards old, inactive tracks. This was not investigated in our work.

Appendix B Data Details

We provide additional details on the datasets used for training, how the data is augmented, and how it is sampled.

Datasets

For the training of the instance segmentation method we use a mix of YouTubeVIS (Yang et al., 2019) and OpenImages (Kuznetsova et al., 2020; Benenson et al., 2019). At first we utilized only YouTubeVIS, but found that it was difficult to obtain good performance. When we adopt the video instance segmentation method MaskTrackRCNN (Yang et al., 2019) with our backbone and detector, we obtain a performance of 19.8 mAP. This is a drop of 10.5 mAP compared to what is reported in their work. The low performance is not too surprising as YouTubeVIS contains 40 classes over 2k videos, *i.e.* only around 50 examples per class. We believe that YOLACT overfits and therefore opt to extend the training set with OpenImages. Indeed, the performance then jumps to 29.3 mAP, similar to the 30.3 reported in their work. We show these results in Table 9.

The YouTubeVIS training set contains 2238 videos, but we keep only those that are of the correct size, 720×1280, and hold out 200 such sequences for validation, giving us a total of 1867 videos for training. We consider each video a sample, and randomly select a single frame. OpenImages contains 237272 images with instance segmentation annotations for most objects of the YouTubeVIS categories. The images vary in resolution, and we resize them to fit 720×1280. Note that OpenImages is only sparsely annotated, containing annotations for only a fraction of all objects in each image. We do not treat the samples from OpenImages differently however.

Table 9 Performance on the YouTubeVIS validation set with YOLACT (Bolya et al., 2019) as detector. YOLACT-OI is trained on YouTubeVIS (Yang et al., 2019) *and* OpenImages (Benenson et al., 2019)

Configuration	mAP
MaskRCNN + MaskTrackRCNN	30.3†
YOLACT + MaskTrackRCNN	19.8
YOLACT-OI + MaskTrackRCNN	29.3
YOLACT-OI + Ours	35.3

† result obtained from (Yang et al., 2019)

Data Augmentation

For each training example, we first uniformly sample height and width from $\mathcal{U}([342\ 608], [720\ 1280])$ and then randomly crop with zero-padding to get an image of size 480×864. We make sure to remove any objects that have disappeared due to the cropping. Last, we randomly flip the image horizontally. During the second training stage, when we train with sequences, we use the same augmentation for all images in a given sequence.

Dataset Sampling

We train the instance segmentation method for 120 epochs, each comprising 7468 samples, *i.e.* four times the size of YouTubeVIS. We randomly sample without replacement from the two datasets, weighting each training sample such that there initially is a 75% chance to select an example from OpenImages. During the second training stage, we train for 150 epochs. In each epoch, we randomly sample without replacement from the 1867 videos of YouTubeVIS. Each sample is a video clip of 10 randomly selected, contiguous frames.

Appendix C Additional Experimental Results

We provide additional results on confidence and true positive tracks, and the sensitivity of λ_3.

Analysis of Confidence and TPs

We supply an additional experiment investigating the distribution of true positive (TP) tracks and false positive (FP) tracks. Tracks are marked as TP or FP in the same way as when computing the VIS-AP performance measure (Yang et al., 2019). Each track is assigned to at most one annotation, and each annotation is assigned to at most one track. A track is only assigned to an annotation if their overlap, in spatiotemporal masks, exceeds some threshold. We adopt an overlap of 0.5.

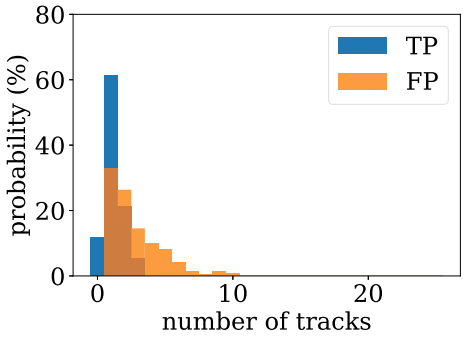

Fig. 10 We provide distributions for the number of true and false positive tracks per sequence, blue and orange, respectively. Each sequence contains 1–5 annotated tracks. In each sequence, our approach predicts tracks that are false positives (Color figure online)

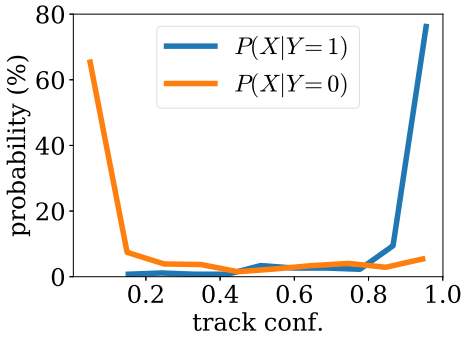

Fig. 11 We show confidence distributions for true positive tracks (blue) and false positive tracks (orange), respectively. Our approach gives false positives low confidence and true positives high confidence. The Pearson correlation coefficient between the track confidence and if it is a true positive is $\rho_{X,Y} = 0.78$ (Color figure online)

Annotated VIS data is required to compute whether a predicted track is a TP or FP. The annotations for the YouTubeVIS validation set have not been made public. We, therefore, split the YouTubeVIS training set into two parts. We retrain our approach using the first part and create statistics from the second part, which comprises 220 sequences. Each of the 220 sequences contains 1 to 5 tracks. We show the distributions in Fig. 10.

We also compute the correlation between the predicted confidence of a track and whether it is a true positive. To this end, let X denote a random variable that is the confidence of a track. Similarly, let Y denote a random binary variable, 1 if the track is a true positive, and 0 if it is a false positive. We compute the distribution of X given Y in Fig. 11. Typically, false positive tracks have lower confidence than true positive tracks. We also compute the Pearson correlation coefficient between the confidence and whether a track is a true positive or a false positive,

$$\rho_{X,Y} = \frac{\mathbb{E}\left[(X - \mu_X)\,(Y - \mu_Y)\right]}{\sigma_X \sigma_Y}. \tag{11}$$

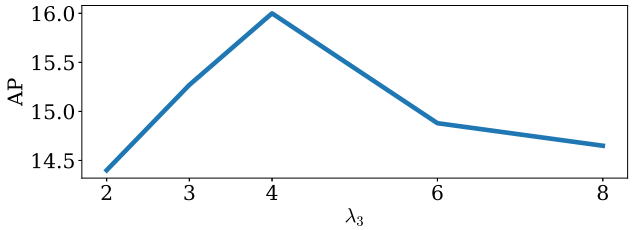

Fig. 12 Investigation of the sensitivity to the hyperparameter λ_3 (see (8)). Performance is on the OVIS validation set. We show the impact around $\lambda_3 = 4$. For lower or higher values $\lambda_3 \in \{2, 3, 5, 6\}$ the accuracy deteriorates (Color figure online)

We find $\rho_{X,Y} = 0.78$ on the 220 held-out sequences. That is, the confidence predicted for a track is highly correlated to whether the track is a true positive.

Sensitivity of λ_3

We do a sensitivity study on the hyperparameter λ_3 (see (8)) to investigate its sensitivity in terms of AP, on the OVIS validation set. In Fig. 12, we show the impact of λ_3 around $\lambda_3 = 4$. For lower or higher values $\lambda_3 \in \{2, 3, 5, 6\}$ the accuracy deteriorates. Reducing the value by 25% leads to a small but noticeable drop in performance, 0.73 AP. Similarly, increasing the parameter by 50% leads to a drop in performance of 1.12 AP.

Appendix D Additional Qualitative Results

We supply additional qualitative results: a qualitative comparison for two of our ablation experiments, examples of the per-frame class and confidence predictions, and success as well as failure cases.

Qualitative Ablation

Section 4.1 of the main paper comprises an ablation study based on quantitative experiments. Here we highlight the difference between the proposed approach and two of the ablation configurations with a qualitative comparison. We compare the proposed approach with the Association from Yang et al. (2019) and the Scoring from Yang et al. (2019) configurations on a video of the YouTubeVIS validation set. The results are shown in Fig. 13. In this sequence, the association from Yang et al. (2019) leads to a high initialization rate of false positive tracks. The scoring from Yang et al. (2019) leads to a high score for a track that is a false positive. Our method in contrast has learnt to predict a low score when tracks are initialized, and later either reinforce or suppress that score. The false positive track in Fig. 13 is quickly sup-

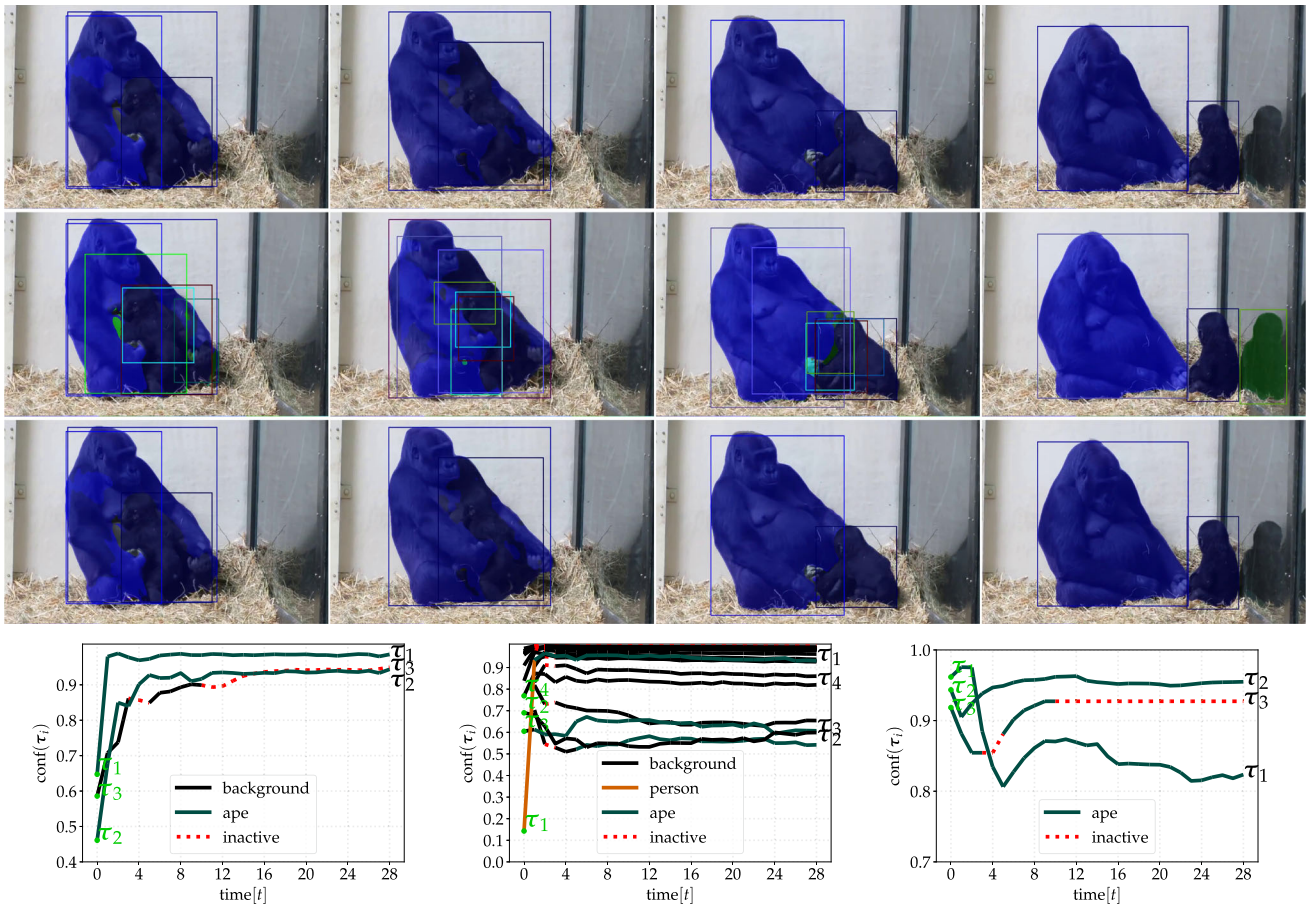

Fig. 13 Here we provide an additional result comparing our approach (first row) with two configurations of our ablation study: i) association from Yang et al. (2019) (second row) and ii) scoring from Yang et al. (2019) (third row). Frames are shown at times $t \in \{3, 4, 7, 25\}$. The last row shows predicted confidence and track identity for each row above. Noisy detections confuse the simple association method, initializing a lot of tracks. For the simple scoring (third row), the track τ_3, false positive covering half the ape on the left, is classified as an ape with high confidence. Our approach (first row) for this sequence can handle the noise and mark τ_3 as background, resulting in a clear sequence of true positive tracks

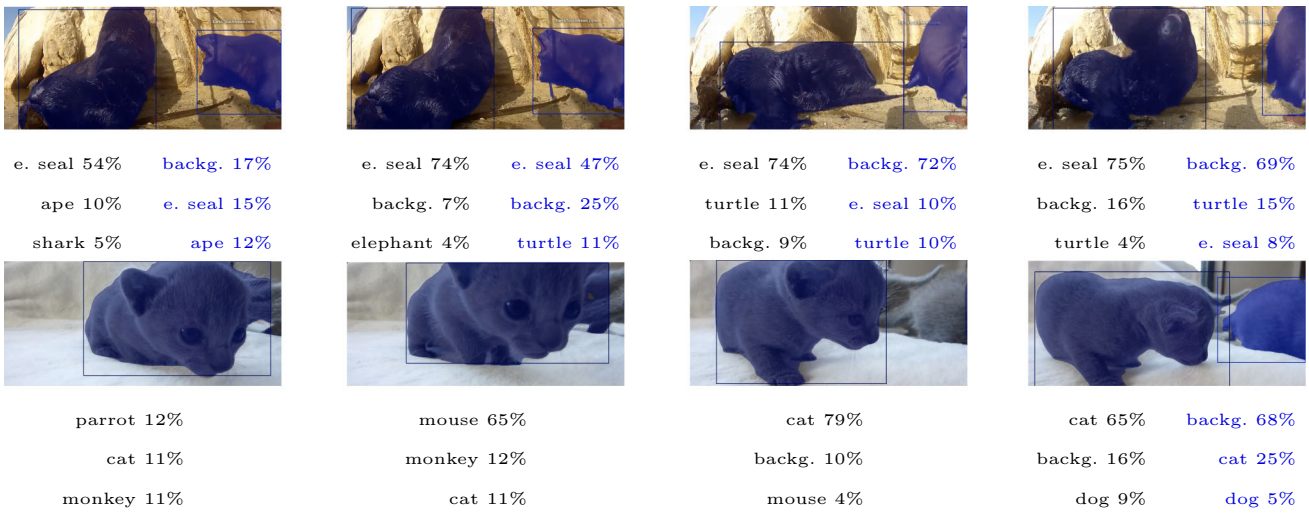

Fig. 14 Our approach predicts the class membership and confidence for each track in each frame. We show one example where the predictions are initially good but then deteriorates (top), and another where it is initially incorrect but is later corrected (bottom). Under each image, we list the top three classes predicted by the model for each track, including the background class

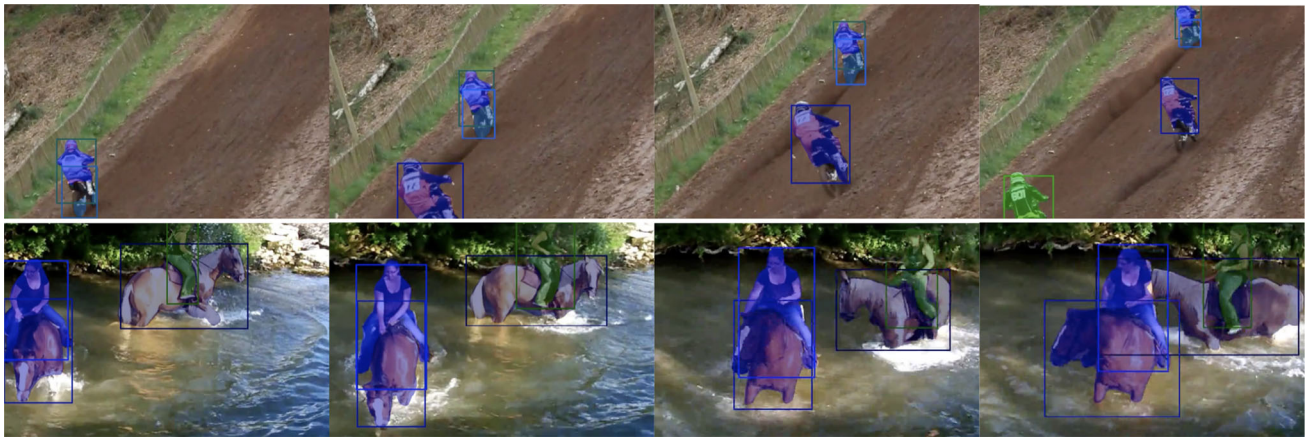

Fig. 15 Two example sequences. Our model successfully performs track initialization and track-detection assignment

Fig. 16 Two failure cases. A penguin is detected and tracked (left). Penguin is not an annotated class in YouTubeVIS. To the right, a depiction of a man is detected and tracked, whereas the model misses the truck on which it is painted

pressed and marked as background. Our approach suppresses noise in the detector and forms two accurate tracks.

Class Membership and Confidence Prediction

In Fig. 14, we show two examples of the class membership and confidence predicted by our approach. In the top row, two earless seals are correctly identified, early on with low confidence but the confidence quickly increases. However, in the middle of the video the prediction deteriorates for one of the earless seals. First, the confidence decreases, but the most probable object category is still earless seal. Then, the model switches the most probable object category to the turtle class.

In the bottom row, the cat is incorrectly identified as a mouse at first. However, after a slight change in viewpoint, our method correctly identifies it as a cat. Like Yang et al. (2019), we report a category as the category most often predicted, for each track.

Success and Failure Cases

Last, in Fig. 15 we show two examples of successful track initialization and track-detection association under the presence of multiple similar objects. Figure 16 shows two failure cases. The model detects and tracks a penguin even though *penguin* is not an annotated class. Furthermore, the model detects and tracks the picture of a human that has been painted on the side of a truck.

References

Athar, A., Mahadevan, S., Ošep, A., Leal-Taixé, L., & Leibe, B. (2020). Stem-seg: Spatio-temporal embeddings for instance segmentation in videos. In Computer Vision - ECCV 2020–16th European Conference, Glasgow, UK, August 23–28, 2020. *Proceedings, Part XI, Lecture Notes in Computer Science, 12356*, 158–177.

Athar, A., Mahadevan, S., Ošep, A., Leal-Taixé, L., & Leibe, B. (2021). A single-stage, bottom-up approach for occluded vis using spatio-temporal embeddings. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV) Workshops*, pp. 3858–3862.

Ba, J.L., Kiros, J.R., & Hinton, G.E. (2016). Layer normalization

Battaglia, P.W., Hamrick, J.B., Bapst, V., Sanchez-Gonzalez, A., Zambaldi, V.F., Malinowski, M., Tacchetti, A., Raposo, D., Santoro, A., Faulkner, R., Gülçehre, Ç., Song, H.F., Ballard, A.J., Gilmer, J., Dahl, G.E., Vaswani, A., Allen, K.R., Nash, C., Langston, V., Dyer, C., Heess, N., Wierstra, D., Kohli, P., Botvinick, M., Vinyals, O., Li, Y., Pascanu, R. (2018). Relational inductive biases, deep learning, and graph networks. CoRR arXiv:1806.01261.

Benenson, R., Popov, S., & Ferrari, V. (2019). Large-scale interactive object segmentation with human annotators. In *CVPR*.

Berg, A., Johnander, J., Durand de Gevigney, F., Ahlberg, J., & Felsberg, M. (2019). Semi-automatic annotation of objects in visual-thermal

video. In *Proceedings of the IEEE International Conference on Computer Vision Workshops*.

Berman, M., Rannen Triki, A., & Blaschko, M.B. (2018). The lovász-softmax loss: A tractable surrogate for the optimization of the intersection-over-union measure in neural networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 4413–4421.

Bertasius, G., & Torresani, L. (2020). Classifying, segmenting, and tracking object instances in video with mask propagation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Rrecognition*, pp. 9739–9748.

Bertasius, G., Torresani, L., & Shi, J. (2018). Object detection in video with spatiotemporal sampling networks. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pp. 331–346.

Bolya, D., Zhou, C., Xiao, F., & Lee, Y.J. (2019). Yolact: Real-time instance segmentation. In *Proceedings of the IEEE International Conference on Computer Vision*, pp. 9157–9166.

Brasó, G., Leal-Taixé, L. (2020). Learning a neural solver for multiple object tracking. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 6247–6257.

Burghardt, T., & Ćalić, J. (2006). Analysing animal behaviour in wildlife videos using face detection and tracking. *IEE Proceedings-Vision, Image and Signal Processing, 153*(3), 305–312.

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

Carion, N., Massa, F., Synnaeve, G., Usunier, N., Kirillov, A., & Zagoruyko, S. (2020). End-to-end object detection with transformers. arXiv preprint arXiv:2005.12872.

Chen, K., Pang, J., Wang, J., Xiong, Y., Li, X., Sun, S., Feng, W., Liu, Z., Shi, J., & Ouyang, W., et al. (2019). Hybrid task cascade for instance segmentation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 4974–4983.

Cho, K., van Merriënboer, B., Bahdanau, D., & Bengio, Y. (2014). On the properties of neural machine translation: Encoder-decoder approaches. *Syntax, Semantics and Structure in Statistical Translation,103*.

Cordts, M., Omran, M., Ramos, S., Rehfeld, T., Enzweiler, M., Benenson, R., Franke, U., Roth, S., & Schiele, B. (2016). The cityscapes dataset for semantic urban scene understanding. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 3213–3223.

Fang, Y., Yang, S., Wang, X., Li, Y., Fang, C., Shan, Y., Feng, B., & Liu, W. (2021). Instances as queries. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, pp. 6910–6919.

Greff, K., Srivastava, R. K., Koutník, J., Steunebrink, B. R., & Schmidhuber, J. (2016). Lstm: A search space odyssey. *IEEE Transactions on Neural Networks and Learning Systems, 28*(10), 2222–2232.

Han, W., Khorrami, P., Paine, T.L., Ramachandran, P., Babaeizadeh, M., Shi, H., Li, J., Yan, S., & Huang, T.S. (2016). Seq-nms for video object detection. arXiv preprint arXiv:1602.08465.

He, K., Zhang, X., Ren, S., & Sun, J. (2016). Deep residual learning for image recognition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 770–778.

Hochreiter, S., & Schmidhuber, J. (1997). Long short-term memory. *Neural Computation, 9*(8), 1735–1780.

Hwang, S., Heo, M., Oh, S. W., & Kim, S. J. (2021). Video instance segmentation using inter-frame communication transformers. *Advances in Neural Information Processing Systems, 34*, 13352–13363.

Izquierdo, R., Quintanar, A., Parra, I., Fernández-Llorca, D., & Sotelo, M. (2019). The prevention dataset: a novel benchmark for prediction of vehicles intentions. In *2019 IEEE Intelligent Transportation Systems Conference (ITSC)*, pp. 3114–3121. IEEE.

Johnander, J., Brissman, E., Danelljan, M., & Felsberg, M. (2021). Video instance segmentation with recurrent graph neural networks. In *DAGM German Conference on Pattern Recognition*, pp. 206–221. Springer.

Kuznetsova, A., Rom, H., Alldrin, N., Uijlings, J., Krasin, I., Pont-Tuset, J., Kamali, S., Popov, S., Malloci, M., Kolesnikov, A., Duerig, T., & Ferrari, V. (2020). The open images dataset v4: Unified image classification, object detection, and visual relationship detection at scale. *IJCV*.

Li, Z., Cao, L., & Wang, H. (2021). Limited sampling reference frame for masktrack r-cnn. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV) Workshops*, pp. 3854–3857.

Li, M., Li, S., Li, L., & Zhang, L. (2021). Spatial feature calibration and temporal fusion for effective one-stage video instance segmentation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 11215–11224.

Liu, Z., Lin, Y., Cao, Y., Hu, H., Wei, Y., Zhang, Z., Lin, S., & Guo, B. (2021). Swin transformer: Hierarchical vision transformer using shifted windows. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, pp. 10012–10022.

Luiten, J., Torr, P., & Leibe, B. (2019). Video instance segmentation 2019: A winning approach for combined detection, segmentation, classification and tracking.

Luiten, J., Zulfikar, I.E., & Leibe, B. (2020). Unovost: Unsupervised offline video object segmentation and tracking. In *2020 IEEE Winter Conference on Applications of Computer Vision (WACV)*, pp. 1989–1998. IEEE.

Meng, D., Chen, X., Fan, Z., Zeng, G., Li, H., Yuan, Y., Sun, L., & Wang, J. (2021). Conditional detr for fast training convergence. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 3651–3660

Murphy, K. P. (2007). Conjugate Bayesian analysis of the gaussian distribution. *def, 1*, 16.

Paszke, A., Gross, S., Chintala, S., Chanan, G., Yang, E., DeVito, Z., Lin, Z., Desmaison, A., Antiga, L., & Lerer, A. (2017). Automatic differentiation in pytorch.

Qi, J., Gao, Y., Hu, Y., Wang, X., Liu, X., Bai, X., Belongie, S., Yuille, A., Torr, P., & Bai, S. (2021). Occluded video instance segmentation: Dataset and challenge. In *Thirty-fifth Conference on Neural Information Processing Systems Datasets and Benchmarks Track (Round 2)* . https://openreview.net/forum?id=IfzTefIU_3j.

Sarlin, P.-E., DeTone, D., Malisiewicz, T., & Rabinovich, A. (2020). Superglue: Learning feature matching with graph neural networks. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 4938–4947.

Scarselli, F., Gori, M., Tsoi, A. C., Hagenbuchner, M., & Monfardini, G. (2008). The graph neural network model. *IEEE Transactions on Neural Networks, 20*(1), 61–80.

T'Jampens, R., Hernandez, F., Vandecasteele, F., & Verstockt, S. (2016). Automatic detection, tracking and counting of birds in marine video content. In *2016 Sixth International Conference on Image Processing Theory, Tools and Applications (IPTA)*, pp. 1–6 . IEEE.

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., et al. (2017). Attention is all you need. *Advances in Neural Information Processing Systems, 30*, 5998–6008.

Voigtlaender, P., Chai, Y., Schroff, F., Adam, H., Leibe, B., & Chen, L.-C. (2019). Feelvos: Fast end-to-end embedding learning for video object segmentation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 9481–9490.

Wang, Y., Xu, Z., Wang, X., Shen, C., Cheng, B., Shen, H., & Xia, H. (2021). End-to-end video instance segmentation with transformers. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 8741–8750.

Weng, X., Wang, Y., Man, Y., & Kitani, K.M. (2020). GNN3DMOT: Graph neural network for 3d multi-object tracking with 2d-3d multi-feature learning. In *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition*, CVPR 2020, Seattle, WA, USA, pp. 6498–6507. IEEE, https://doi.org/10.1109/CVPR42600.2020.00653.

Wojke, N., Bewley, A., & Paulus, D. (2017). Simple online and realtime tracking with a deep association metric. In *2017 IEEE International Conference on Image Processing (ICIP)*, pp. 3645–3649 . IEEE.

Wu, J., Cao, J., Song, L., Wang, Y., Yang, M., & Yuan, J. (2021). Track to detect and segment: An online multi-object tracker. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 12352–12361.

Yang, L., Fan, Y., & Xu, N. (2019). Video instance segmentation. In *Proceedings of the IEEE International Conference on Computer Vision*, pp. 5188–5197.

Yang, S., Fang, Y., Wang, X., Li, Y., Fang, C., Shan, Y., Feng, B., & Liu, W. (2021). Crossover learning for fast online video instance segmentation. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, pp. 8043–8052.

Yang, L., Wang, Y., Xiong, X., Yang, J., & Katsaggelos, A.K. (2018). Efficient video object segmentation via network modulation. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*

Yu, F., Chen, H., Wang, X., Xian, W., Chen, Y., Liu, F., Madhavan, V., & Darrell, T. (2020). Bdd100k: A diverse driving dataset for heterogeneous multitask learning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 2636–2645.

Zhang, X.-Y., Wu, X.-J., Zhou, X., Wang, X.-G., & Zhang, Y.-Y. (2008). Automatic detection and tracking of maneuverable birds in videos. In *2008 International Conference on Computational Intelligence and Security*, vol. 1, pp. 185–189. IEEE.

Zhu, X., Su, W., Lu, L., Li, B., Wang, X., & Dai, J. (2020). Deformable detr: Deformable transformers for end-to-end object detection. In *International Conference on Learning Representations*.