Video Instance Segmentation with Recurrent Graph Neural Networks

Joakim Johnander[1,2](✉), Emil Brissman[1,3], Martin Danelljan[4], and Michael Felsberg[1,5]

[1] Computer Vision Laboratory, Department of Electrical Engineering, Linköping University, Linköping, Sweden
joakim.johnander@liu.se
[2] Zenseact, Gothenburg, Sweden
[3] Saab, Bröderna Ugglas gata, 582 54 Linköping, Sweden
[4] Computer Vision Lab, ETH Zürich, Zürich, Switzerland
[5] School of Engineering, University of KwaZulu-Natal, Durban, South Africa

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, operating at over 25 FPS, outperforms previous video real-time methods. We further conduct detailed ablative experiments that validate the different aspects of our approach.

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 [14,32], data annotation [4,19], and biology [10,26,33]. In contrast to image instance segmentation, the temporal aspect of its video counterpart poses several additional challenges. Preserving correct instance identities in each frame 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 include wild camera motion and severe background clutter.

Supplementary Information The online version contains supplementary material available at https://doi.org/10.1007/978-3-030-92659-5_13.

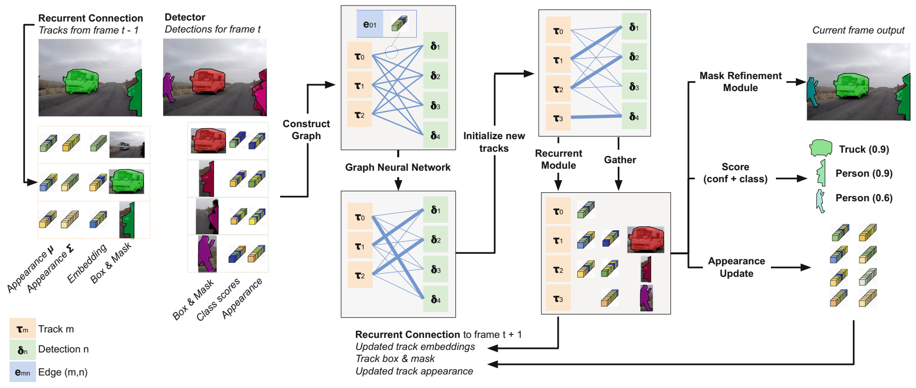

Fig. 1. Illustration of the proposed approach. An instance segmentation method is applied to each frame. The set of detections is, together with a maintained memory of tracks, used to construct a graph. Each node and each edge is represented with an embedding. These are processed by a graph neural network and directly used to predict assignment and track initialization (see Sect. 3.1). The track embeddings are further processed by a recurrent module, producing final track embeddings (Sect. 3.3). These are used to predict track confidences and class memberships (Sect. 3.4), masks (Sect. 3.4), and updated appearance descriptors (Sect. 3.2). Last, the final track embeddings are propagated to the next frame via the recurrent connection.

Prior works have taken inspiration from the related areas of multiple object tracking, video object detection, instance segmentation, and video object segmentation [1,6,30]. Most methods adopt the tracking-by-detection paradigm popular in multiple object tracking [9]. In this paradigm, an instance segmentation method provides detections in each frame, reducing the task to the formation of *tracks* from these detections. Given a set of already initialized tracks, one must determine for each detection whether it belongs to one of the tracks, is a false positive, or if it should initialize a new track. Most approaches [6,11,21,30] 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.

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 [11,30]. (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 [6,11,21,30]. 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.

We address these two drawbacks by proposing a novel spatiotemporal learning framework for video instance segmentation that closely models the inference stage during training. Our network proceeds frame by frame, and is in each frame asked to create tracks, associate detections to tracks, and score existing tracks. We use this formulation 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 an existing track and a detection, the probability that the track and the detection correspond to the same instance. 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.

Contributions: Our main contributions 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, 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 permits 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 [30].

2 Related Work

The video instance segmentation (VIS) problem was introduced by Yang *et al.* [30]. With it, they proposed several simple and straightforward approaches to tackle the task. They follow the tracking-by-detection paradigm and first apply an instance segmentation method to provide detections in each frame, and then form tracks based on these detections. They experiment with several approaches to matching different detections, such as mask propagation with a video object segmentation method [27]; application of a multiple object tracking method [29] in which the image-plane bounding boxes are Kalman filtered and targets are re-detected with a learnt re-identification mechanism; and similarity learning of instance-specific appearance descriptors [30]. Additionally, they experiment with the offline temporal filtering proposed in [16].

Cao *et al.* [11] propose to improve the underlying instance segmentation method, obtaining both better performance and computational efficiency. Luiten *et al.* [21] 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 [22]. Bertasius *et al.* [6] also utilize a more powerful instance segmentation method [7], and propose a novel mask propagation method based on deformable convolutions. Both [21] and [6] achieve strong performance, but at a very high computational cost.

All of these approaches follow the tracking-by-detection paradigm and try various ways to improve the underlying instance segmentation method or the association of detections. The latter relies mostly on heuristics [21,30] and is often not end-to-end trainable. Furthermore, the track scoring step, where the class and confidence is predicted, has received little attention and is in existing approaches calculated with a majority vote and an averaging operation, as pointed out in the introduction. The work of Athar *et al.* [1] instead proposes an end-to-end trainable approach that is trained to predict instance center heatmaps and an embedding for each pixel. A track is constructed from strong responses in the heatmap. The embedding at that location is matched with the embeddings of all other pixels, and sufficiently similar pixels are assigned to that track.

Our approach is closely related to two works on multiple object tracking (MOT) [9,28] and a work on feature matching [25]. 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 is to be provided for the tracks. This is reflected in the way [9,25,28] utilizes 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) predict tentative single-image instance segmentations, (ii) associate detections to existing tracks, (iii) initialize new tracks, (iv) score existing tracks, (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, permitting 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

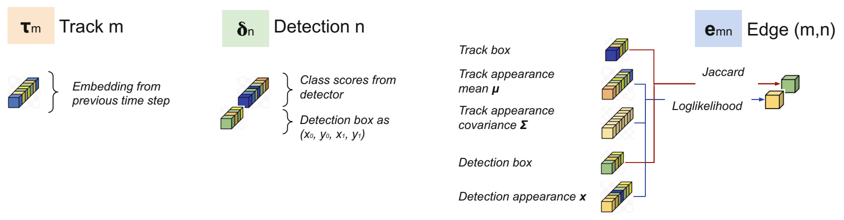
τ_m Track m
Embedding from previous time step
δ_n Detection n
Class scores from detector
Detection box as (x_0, y_0, x_1, y_1)
Track box
Track appearance mean $\boldsymbol{\mu}$
Track appearance covariance $\boldsymbol{\Sigma}$
Detection box
Detection appearance $\boldsymbol{x}$
e_{mn} Edge (m,n)
Jaccard
Loglikelihood

Fig. 2. Overview of the node and edge initialization during graph construction.

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.

Motivation. Most existing methods [11,21,30] 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 scenarios 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 [11,30] achieve this observation 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.

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 was 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 (i) the spatial similarity between them, based on the Jaccard index between their bounding boxes (see [30]); and (ii) the appearance similarity between them, as described in Sect. 3.2. This construction is illustrated in Fig. 2. Further, 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 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. 1.

GNN-Based Association. The idea is to propagate information between the different embeddings in a learnable way, providing us with updated embeddings that we can directly use to predict the quantities needed for video instance segmentation. To this end we use layers that perform updates of the form

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

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

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

Here, i enumerates the network layers. The functions f_i^e, f_i^τ, and f_i^δ are linear layers followed by a ReLU activation. The gating functions g_i^τ and g_i^δ are multi-layer perceptrons ending with the logistic sigmoid. $[\cdot, \cdot]$ denotes concatenation.

The aforementioned formulation has the structure of a Graph Neural Network (GNN) block [2], with both τ_m and δ_n as nodes, and e_{mn} as edges. These layers permit information exchange between the embeddings. The layer deviates 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 Brasó *et al.* [9] 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 residual blocks [17] in which there is no information exchange between different graph elements. That is, for these blocks the $f_i^{\cdot}$ rely only on their first arguments. Note that these blocks use fully connected layers instead of 2D convolutions. The 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.

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

$$\Pr(m \,\text{matches}\, n) = \text{sigmoid}(w \cdot e_{mn} + b). \tag{2}$$

If the probability is high, they are considered to match and the track will obtain the segmentation of that detection. 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, we initialize a new track 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.

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, we add an appearance network, comprising a few convolutional layers, and apply it to feature maps of the backbone ResNet [17]. The output of the appearance network is pooled with the masks provided by the detections, resulting in an appearance descriptor for each detection. The tracks gather appearance from the detections and over time construct an appearance model of that track. 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 us to discriminate between visually or semantically similar instances.

Our initial experiments of integrating appearance information directly into the GNN, similar to [9,25,28], did not lead to noticeable improvement. This is likely due to differences between the problems. The video instance segmentation problem is fairly unconstrained: there is significant variation in scenes and objects considered, and compared to its variation, there are quite few labelled training sequences available. In contrast, multiple object tracking typically works with a single type of scene or a single category of objects and feature matching 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. Each track models its appearance as a multidimensional Gaussian distribution with diagonal covariance. When the track is initialized, we use the appearance vector of the initializing detection as mean μ and a fixed covariance Σ. We feed appearance information into the GNN via the track-detection edges. The edge between track m and detection n is initialized with the loglikelihood of the detection appearance given the track distribution. The GNN is able to utilize this information when calculating the matching probability of each track-detection pair. Afterwards, the appearance (μ, Σ) of each track is updated with the appearance x of the best matching detection. The update is based on the Bayesian update of a Gaussian under a conjugate prior. We use a normal-inverse-chi-square prior [23],

$$\mu^+ = \kappa x + (1-\kappa)\mu \ , \tag{3a}$$

$$\Sigma^+ = \nu\tilde{\Sigma} + (1-\nu)\Sigma + \frac{\kappa(1-\nu)}{\kappa+\nu}(x-\mu)^2 \ . \tag{3b}$$

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 based on the track embedding, permitting the network to learn a good update strategy.

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. 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) [18] or Gated Recurrent Unit [13]. 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 [13,15,18].

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, denote the output of the GNN as

$$\{\tilde{\tau}_m^t\}, \{\tilde{\delta}_n^t\}, \{\tilde{e}_{mn}^t\} = \mathrm{GNN}(\{\tau_m^{t-1}\}, \{\delta_n^t\}, \{e_{mn}^t\}) \ , \tag{4}$$

where superscript t denotes time. Next, we feed each track embedding $\tilde{\tau}_m^t$ through the LSTM system of gates

$$\begin{aligned}
\alpha_m^{\text{forget}} &= \sigma(h^{\text{forget}}(\tilde{\tau}_m^t)) \ , && \text{(5a)}\\
\alpha_m^{\text{input}} &= \sigma(h^{\text{input}}(\tilde{\tau}_m^t)) \ , && \text{(5b)}\\
\alpha_m^{\text{output}} &= \sigma(h^{\text{output}}(\tilde{\tau}_m^t)) \ , && \text{(5c)}\\
\tilde{c}_m^t &= \tanh(h^{\text{cell}}(\tilde{\tau}_m^t)) \ , && \text{(5d)}\\
c_m^t &= \alpha_m^{\text{forget}} \odot c_i^{t-1} + \alpha_m^{\text{input}} \odot \tilde{c}_m^t \ , && \text{(5e)}\\
\tau_m^t &= \alpha_m^{\text{output}} \odot \tanh(c_m^t) \ . && \text{(5f)}
\end{aligned}$$

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.

3.4 VIS Output Prediction

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 [6,11,21,30] 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 $\{\tau_m\}_m$ 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.

Segmentation. In each frame, we report a segmentation. This segmentation is based on both the track embeddings and the masks provided with the detections. Each track that matches sufficiently well with a detection claims the mask of that detection. This mask together with the track embedding are then fed through a small CNN that reweights and refines the mask. This permits our model to not assign pixels to tracks that it believes are false positives.

3.5 Training

We train the network by feeding a sequence of T frames through it 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

$$\mathbf{y}_t^{\text{match}} \in [0,1]^{M_t \times N_t} \ , \tag{6a}$$
$$\mathbf{y}_t^{\text{init}} \in [0,1]^{N_t} \ , \tag{6b}$$
$$\mathbf{y}_t^{\text{score}} \in [0,1]^{M_{t+1} \times C} \ , \tag{6c}$$
$$\mathbf{y}_t^{\text{seg}} \in [0,1]^{M_{t+1} \times H \times W} \ . \tag{6d}$$

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 (6) permits the model to conduct video instance segmentation. We penalize each with a corresponding loss component

$$\mathcal{L} = \lambda^1 \mathcal{L}^{\text{score}} + \lambda^2 \mathcal{L}^{\text{seg}} + \lambda^3 \mathcal{L}^{\text{match}} + \lambda^4 \mathcal{L}^{\text{init}} \ . \tag{7}$$

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.

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.

$\boldsymbol{\mathcal{L}}^{\textbf{match}}$ 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.

$\boldsymbol{\mathcal{L}}^{\textbf{init}}$ 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.

$\mathcal{L}^{\mathbf{score}}$ 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.
$\mathcal{L}^{\mathbf{seg}}$ is the Lovasz loss [5]. 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.

4 Experiments

We evaluate the proposed approach for video instance segmentation on YouTube-VIS [30] (2019), a benchmark comprising 40 object categories in 2k training videos and 300 validation videos. Performance is measured in terms of video mean average precision (mAP). We first provide qualitative results, showing that the proposed neural network learns to tackle the video instance segmentation problem. Next, we quantitatively compare to the state-of-the-art. Last, we analyze the different components and aspects of our approach in an ablation study.

4.1 Implementation Details

We implement the proposed approach in PyTorch [24] and will make code available upon publication. We aim for real-time performance and therefore select YOLACT [8] as base instance segmentation method. We use the implementation publicly provided by the authors. The detector and our ResNet50 [17] backbone are initialized with weights provided with the YOLACT implementation. We fine-tune the detector on images from YouTubeVIS and OpenImages [3,20] 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, each 10 frames sampled randomly from YouTubeVIS. During training, 200 sequences of YouTubeVIS are held-out for hyperparameter selection. For additional model and training details, see Supplementary material.

4.2 Qualitative Results

In Fig. 3 we show the output of the detector and the tracks predicted by our approach. The detector may provide noisy class predictions. Our model learns to filter these predictions and accurately predict the correct class. When the detector fails to detect an object, our approach pauses the corresponding track until the detector again finds the object. 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 Supplementary material.

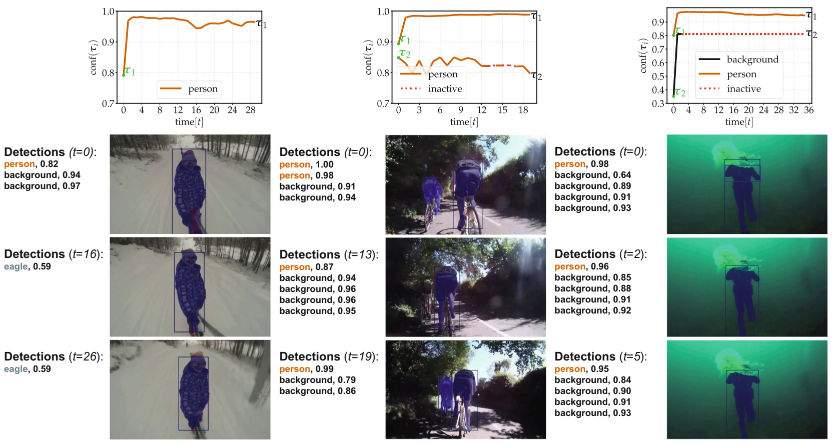

Fig. 3. Track score plots (top) and detections (3 bottom rows) for three videos. The plot colour is the ground-truth class for that track and the value is the confidence for that class, ideally 1.00. In the left video, the detector makes noisy class predictions, but our approach learns to filter this 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.

4.3 Quantitative Comparison

Next, we compare our approach to the state-of-the-art, including the baselines proposed in [30]. The results are shown in Table 1a. Our approach, running at 30 fps, outperforms all near real-time methods. DeepSORT [29], which relies on Kalman-filtering the bounding boxes and a learnt appearance descriptor used for re-identification, obtains an mAP score of 26.1. MaskTrack R-CNN [30] gets a score of 30.3. SipMask [11] improves MaskTrack R-CNN by changing its detector and reach a score of 33.7. Using a ResNet50 backbone, we run at similar speed and outperform all three methods with an absolute gain of 9.2, 5.0, and 1.6 respectively.

While [6,21] obtain higher mAP, those methods are more than a magnitude slower, and thus infeasible for real-time applications or for processing large amounts of data. STEm-Seg [1] reports results using both a ResNet50 and a ResNet101 backbone. We show a gain of 4.7 mAP with ResNet50. We also try with a ResNet101 backbone, retraining our base detector and approach. This leads to a performance of 37.7 mAP, an absolute gain of 3.1.

4.4 Ablation Study

Last, we analyze the different aspects of the proposed approach, with results provided in Table 1b. For additional experiments, see Supplementary material.

Table 1. (a) State-of-the-art comparison on the YouTubeVIS validation dataset [30]. The proposed approach outperforms all near real-time approaches. †: No speed reported in [21] or [6], but each utilize components ([22] and [12]) with a reported speed of 1 fps and 2 fps respectively. (b) Performance under different configurations on the YouTube-VIS validation set. 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 video instance segmentation problem using the mechanism proposed by Yang et al. [11,30].

Method	fps	mAP
OSMN MaskProp [31]		23.4
FEELVOS [27]		26.9
IoUTracker+ [30]		23.6
OSMN [31]		27.5
DeepSORT [29]		26.1
SeqTracker [30]		27.5
MaskTrack R-CNN [30]	20	30.3
SipMask [11]	30	32.5
SipMask ms-train [11]	30	33.7
STEm-Seg ResNet50 [1]		30.6
STEm-Seg ResNet101 [1]	7	34.6
VIS2019 Winner [21]	$< 1^{\dagger}$	**44.8**
MaskProp [6]	$< 2^{\dagger}$	**46.6**
Ours (ResNet50)	30	*35.3*
Ours (ResNet101)	25	**37.7**

(a)

Configuration	mAP
Our final approach	35.3
No GNN	28.6
No LSTM-like gating	Diverges
Simple recurrent gate	31.5
No appearance	34.5
Appearance baked into embedding	34.4
Association from [11,30]	29.2
Scoring from [11,30]	31.5

(b)

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 6.7 drop in mAP, demonstrating the importance of our GNN.

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.

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.

Association or Scoring from [30]**.** 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 [11,30]. This leads to performance drops of 6.1 and 3.8 mAP respectively.

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.

Acknowledgement. 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 Swedish National Infrastructure for Computing (SNIC), partially funded by the Swedish Research Council through grant agreement no. 2018-05973.

References

1. Athar, A., Mahadevan, S., Ošep, A., Leal-Taixé, L., Leibe, B.: STEm-Seg: spatio-temporal embeddings for instance segmentation in videos. In: Vedaldi, A., Bischof, H., Brox, T., Frahm, J.-M. (eds.) ECCV 2020. LNCS, vol. 12356, pp. 158–177. Springer, Cham (2020). https://doi.org/10.1007/978-3-030-58621-8_10
2. Battaglia, P.W., et al.: Relational inductive biases, deep learning, and graph networks. CoRR abs/1806.01261 (2018)
3. Benenson, R., Popov, S., Ferrari, V.: Large-scale interactive object segmentation with human annotators. In: CVPR (2019)
4. Berg, A., Johnander, J., Durand de Gevigney, F., Ahlberg, J., Felsberg, M.: Semi-automatic annotation of objects in visual-thermal video. In: Proceedings of the IEEE International Conference on Computer Vision Workshops (2019)
5. Berman, M., Rannen Triki, A., Blaschko, M.B.: 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 (2018)
6. Bertasius, G., Torresani, L.: Classifying, segmenting, and tracking object instances in video with mask propagation. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, pp. 9739–9748 (2020)
7. Bertasius, G., Torresani, L., Shi, J.: Object detection in video with spatiotemporal sampling networks. In: Ferrari, V., Hebert, M., Sminchisescu, C., Weiss, Y. (eds.) ECCV 2018. LNCS, vol. 11216, pp. 342–357. Springer, Cham (2018). https://doi.org/10.1007/978-3-030-01258-8_21

8. Bolya, D., Zhou, C., Xiao, F., Lee, Y.J.: YOLACT: real-time instance segmentation. In: Proceedings of the IEEE International Conference on Computer Vision, pp. 9157–9166 (2019)
9. Brasó, G., Leal-Taixé, L.: Learning a neural solver for multiple object tracking. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, pp. 6247–6257 (2020)
10. Burghardt, T., Ćalić, J.: Analysing animal behaviour in wildlife videos using face detection and tracking. IEE Proc.-Vis. Image Signal Process. **153**(3), 305–312 (2006)
11. Cao, J., Anwer, R.M., Cholakkal, H., Khan, F.S., Pang, Y., Shao, L.: SipMask: spatial information preservation for fast image and video instance segmentation. In: Vedaldi, A., Bischof, H., Brox, T., Frahm, J.-M. (eds.) ECCV 2020. LNCS, vol. 12359, pp. 1–18. Springer, Cham (2020). https://doi.org/10.1007/978-3-030-58568-6_1
12. Chen, K., et al.: Hybrid task cascade for instance segmentation. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pp. 4974–4983 (2019)
13. Cho, K., van Merriënboer, B., Bahdanau, D., Bengio, Y.: On the properties of neural machine translation: encoder-decoder approaches. Syntax, Semantics and Structure in Statistical Translation, p. 103 (2014)
14. Cordts, M., et al.: The cityscapes dataset for semantic urban scene understanding. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pp. 3213–3223 (2016)
15. Greff, K., Srivastava, R.K., Koutník, J., Steunebrink, B.R., Schmidhuber, J.: LSTM: a search space odyssey. IEEE Trans. Neural Netw. Learn. Syst. **28**(10), 2222–2232 (2016)
16. Han, W., et al.: SEQ-NMS for video object detection. arXiv preprint arXiv:1602.08465 (2016)
17. He, K., Zhang, X., Ren, S., Sun, J.: Deep residual learning for image recognition. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pp. 770–778 (2016)
18. Hochreiter, S., Schmidhuber, J.: Long short-term memory. Neural Comput. **9**(8), 1735–1780 (1997)
19. Izquierdo, R., Quintanar, A., Parra, I., Fernández-Llorca, D., Sotelo, M.: The prevention dataset: a novel benchmark for prediction of vehicles intentions. In: 2019 IEEE Intelligent Transportation Systems Conference (ITSC), pp. 3114–3121. IEEE (2019)
20. Kuznetsova, A., et al.: The open images dataset v4: unified image classification, object detection, and visual relationship detection at scale. IJCV **128**, 1956–1981 (2020)
21. Luiten, J., Torr, P., Leibe, B.: Video instance segmentation 2019: a winning approach for combined detection, segmentation, classification and tracking. In: Proceedings of the IEEE International Conference on Computer Vision Workshops (2019)
22. Luiten, J., Zulfikar, I.E., Leibe, B.: UnOVOST: unsupervised offline video object segmentation and tracking. In: 2020 IEEE Winter Conference on Applications of Computer Vision (WACV), pp. 1989–1998. IEEE (2020)
23. Murphy, K.P.: Conjugate Bayesian analysis of the Gaussian distribution. def $1(2\sigma 2)$, 16 (2007)
24. Paszke, A., et al.: Automatic differentiation in PyTorch (2017)

25. Sarlin, P.E., DeTone, D., Malisiewicz, T., Rabinovich, A.: SuperGlue: learning feature matching with graph neural networks. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, pp. 4938–4947 (2020)
26. T'Jampens, R., Hernandez, F., Vandecasteele, F., Verstockt, S.: 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 (2016)
27. Voigtlaender, P., Chai, Y., Schroff, F., Adam, H., Leibe, B., Chen, L.C.: 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 (2019)
28. Weng, X., Wang, Y., Man, Y., Kitani, K.M.: 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, June 13–19, 2020, pp. 6498–6507. IEEE (2020). https://doi.org/10.1109/CVPR42600.2020.00653
29. Wojke, N., Bewley, A., Paulus, D.: Simple online and realtime tracking with a deep association metric. In: 2017 IEEE International Conference on Image Processing (ICIP), pp. 3645–3649. IEEE (2017)
30. Yang, L., Fan, Y., Xu, N.: Video instance segmentation. In: Proceedings of the IEEE International Conference on Computer Vision, pp. 5188–5197 (2019)
31. Yang, L., Wang, Y., Xiong, X., Yang, J., Katsaggelos, A.K.: Efficient video object segmentation via network modulation. In: The IEEE Conference on Computer Vision and Pattern Recognition (CVPR) (2018)
32. Yu, F., et al.: 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 (2020)
33. Zhang, X.Y., Wu, X.J., Zhou, X., Wang, X.G., Zhang, Y.Y.: Automatic detection and tracking of maneuverable birds in videos. In: 2008 International Conference on Computational Intelligence and Security, vol. 1, pp. 185–189. IEEE (2008)