Learning Hierarchical Embedding for Video Instance Segmentation

Qin, Z.; Lu, X.; Nie, X.; Zhen, X.; Yin, Y.

DOI
10.1145/3474085.3475342

Publication date
2021

Document Version
Final published version

Published in
MM '21

License
Unspecified

Learning Hierarchical Embeddings for Video Instance Segmentation

Zheyun Qin
School of Software, Shandong University
Jinan, China
zyqin@mail.sdu.edu.cn

Xiankai Lu[*]
School of Software, Shandong University
Jinan, China
carrierlxk@gmail.com

Xiushan Nie
School of Computer Science and Technology, Shandong Jianzhu University
Jinan, China
niexsh@hotmail.com

Xiantong Zhen
VIS Lab, University of Amsterdam
Amsterdam, Netherlands
x.zhen@uva.nl

Yilong Yin[*]
School of Software, Shandong University
Jinan, China
ylyin@sdu.edu.cn

ABSTRACT

In this paper, we address video instance segmentation using a new generative model that learns effective representations of the target and background appearance. We propose to exploit hierarchical structural embedding over spatio-temporal space, which is compact, powerful, and flexible in contrast to current tracking-by-detection methods. Specifically, our model segments and tracks instances across space and time in a single forward pass, which is formulated as hierarchical embedding learning. The model is trained to locate the pixels belonging to specific instances over a video clip. We firstly take advantage of a novel mixing function to better fuse spatio-temporal embeddings. Moreover, we introduce normalizing flows to further improve the robustness of the learned appearance embedding, which theoretically extends conventional generative flows to a factorized conditional scheme. Comprehensive experiments on the video instance segmentation benchmark, *i.e.*, YouTube-VIS, demonstrate the effectiveness of the proposed approach. Furthermore, we evaluate our method on an unsupervised video object segmentation dataset to demonstrate its generalizability.

CCS CONCEPTS

• **Computer systems organization** → **Embedded systems**; *Redundancy*; Robotics; • **Networks** → Network reliability.

KEYWORDS

Video Instance Segmentation, Mixture Model, Normalizing Flows, Embedding Learning

ACM Reference Format:
Zheyun Qin, Xiankai Lu, Xiushan Nie, Xiantong Zhen, and Yilong Yin. 2021. Learning Hierarchical Embeddings for Video Instance Segmentation. In *Proceedings of the 29th ACM Int'l Conference on Multimedia (MM '21), Oct. 20–24, 2021, Virtual Event, China.* ACM, New York, NY, USA, 9 pages. https://doi.org/10.1145/3474085.3475342

1 INTRODUCTION

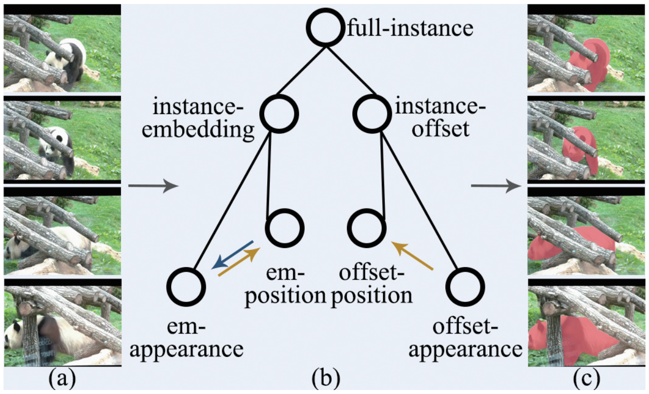

Figure 1: Illustration of our hierarchical generative pipeline. (a) Input video clip. (b) The hierarchical embedding learning for instance inference, where / represents mixing function, → and → represent position information transfer and refinement, respectively. (c) The results.

Video instance segmentation (VIS) involves tracking, segmentation, and classifying all instances in a video sequence [44, 47]. Compared with classical video object segmentation [23–26, 40–42], VIS faces greater challenges as follows: 1) There is no annotation at the beginning of the video, so the VIS algorithm must identify and associate all instances automatically. 2) The low quality of videos caused by low resolution and motion blur.

To tackle these challenges, most mainstream VIS methods follow a discriminative, tracking-by-detection paradigm. Typically, such methods employ an object detection network (*e.g.*, Mask-RCNN) to generate a proposal for each frame and associate them over consecutive frames with motion cues (*e.g.*, optical flow) or appearance

[*] Corresponding author

MM '21, October 20–24, 2021, Virtual Event, China.

ACM ISBN 978-1-4503-8651-7/21/10…$15.00
https://doi.org/10.1145/3474085.3475342

cues (pixel matching). Furthermore, some efforts have focused on designing an extra re-identification module to handle occlusion and out-view cases. Although these methods have yielded distinct performance gains, such multiple-stage strategies have several disadvantages: 1) They require designing and finetuning a separate model in each independent step, and are not end-to-end trainable. These results in a costly and cumbersome solution. 2) The natural distractors in videos, such as low resolution and motion blur, often cause the detectors to fail to locate new instances.

To address the shortcomings of these prior methods, we propose a new paradigm for bottom-up, generative, and end-to-end video instance segmentation. In contrast to previous methods, our network learns a generative probabilistic model of the instance feature distributions hierarchically. Moreover, we integrate a factorized conditional flow to refine the feature distribution to obtain a more robust representation of each instance.

In our generative appearance module, both the model inference and prediction stages are fully differentiable. This ensures that the entire segmentation pipeline can be trained end-to-end, which is not the case with previous discriminative methods [3, 6, 28, 47] invoking step-by-step operations. Three crucial techniques are exploited to deliver our compact and powerful VIS solution:

- *Generative Instance Representation:* We leverage the representational power of spatio-temporal embeddings to distinguish each object instance in a video in a bottom-up fashion. We learn the embeddings in a category-agnostic setting, such that pixels belonging to the same object instance across the spatio-temporal volume are mapped to a single distribution in the embedding space. Hence, we can infer object instances by simply assigning pixels to their respective distribution.
- *Coarse-to-Fine Inference:* We implement the whole instance inference in a coarse-to-fine manner. We locate the coarse position of each instance by learning spatio-temporal embeddings with the mixing function. Then, we refine the inference status using appearance embedding-based normalizing flows.
- *An End-to-End Trainable Framework:* Current top-down VIS methods independently implement instance inference. This breaks the end-to-end pipeline and leads to suboptimal results. We implemented the entire VIS inference with a differentiable neural network. The solution is neat and compacts with high-speed inference speed.

To summarize, our contributions are as follows: (i) We exploit a hierarchical generative pipeline for VIS. The entire framework is elegant and effective. (ii) Compared to discriminative VIS networks that require multiple networks with careful parameter tuning, our network is trained stably with a single network for spatio-temporal embeddings as well as parameter learning simultaneously. (iii) For the first time in this field, conditional flow is used to estimate the instance appearance distribution.

We perform extensive experiments on a representative VIS dataset: YouTube-VIS$_{19}$ [19]. The experimental results show that our approach outperforms state-of-the-art approaches. We also perform experiments on the famous unsupervised video object segmentation dataset and again confirmed the superiority of our model.

2 RELATED WORKS

2.1 Video Instance Segmentation

VIS not only requires instance segmentation of individual frames, but also the tracking of instances across frames. Current top-leading deep learning-based VIS models are mainly built on two paradigms: *multiple-stage* and *single-stage*.

Multiple-stage approaches [3, 6, 10, 12, 16, 19, 21, 28, 29, 47] typically segment the instances of each frame and then correlate them. MaskTrack R-CNN [47] was the first attempt that extended the original Mask R-CNN [14] with a tracking branch to implement single-frame instance segmentation and inter-frame tracking. This tracking-by-segmentation pipeline has been widely used in previous studies [3, 6, 28]. For instance, Luiten *et.al.* [28] further introduced a classification branch into MaskTrack and designed an ensemble approach to solving these separate subproblems. Meanwhile, MaskProp [3] adapted an extra mask propagation branch that generates clip-level instance tracks densely for each frame and aggregated them to produce video-level results. Recently, SipMask [6] improved Mask R-CNN [47] by generating spatial coefficients for each instance with a novel lightweight spatial preservation on its detector. Similarly, Lin *et.al.* [19] added a modified variational autoencoder to facilitate the training of the Mask R-CNN. All of these approaches follow the discriminant paradigm and require the design and tuning of a separate model to improve overall performance. Therefore, they rely heavily on image-level instance segmentation models and complex human-designed rules to associate instances.

For single-stage VIS, STEm-Seg [2] exploited a video clip as a single 3D spatio-temporal volume, enhanced the feature representation of spatio-temporal embeddings, and then separated object instances by clustering learned embeddings. It may be noted that the above approaches either rely on complex training rules to associate instances or require multiple steps to generate and optimize the masks iteratively. In contrast, we aim to build a simple and end-to-end trainable VIS framework.

2.2 Normalizing Flows

Unlike generative adversarial networks (GANs) [13] and variational auto-encoders (VAEs) [18], flow-based generative models [8, 9, 17] build a series of invertible transformations and directly optimize the negative log-likelihood of data distribution using standard stochastic gradient descent (SGD) optimization.

As a representative flow method, normalizing flows have been explored for several vision tasks. Ardizzone *et.al.* [1] and Winkler *et.al.* [45] addressed the task of natural image generation guided by a conditioning input by concatenating the encoded conditioning variable in the affine coupling layers to guide the generation of diverse images with high realism. SRFlow [27] investigated a way to utilize normalizing flows to learn strong image posterior to generate super-resolution images. Moreover, for image denoising and restoration, Zanfir *et.al.* [50] designed different normalizing flows-based prior representations, which were used for the first time in modeling a 3D human pose. Recently, C-flow [34] and Pointflow [46] integrated normalizing flows into 3D point clouds with considerable possibilities for multimodal data modeling. Although we also employ the normalizing flows paradigm owing to its theoretically appealing properties, our method differs from these previous approaches. Our

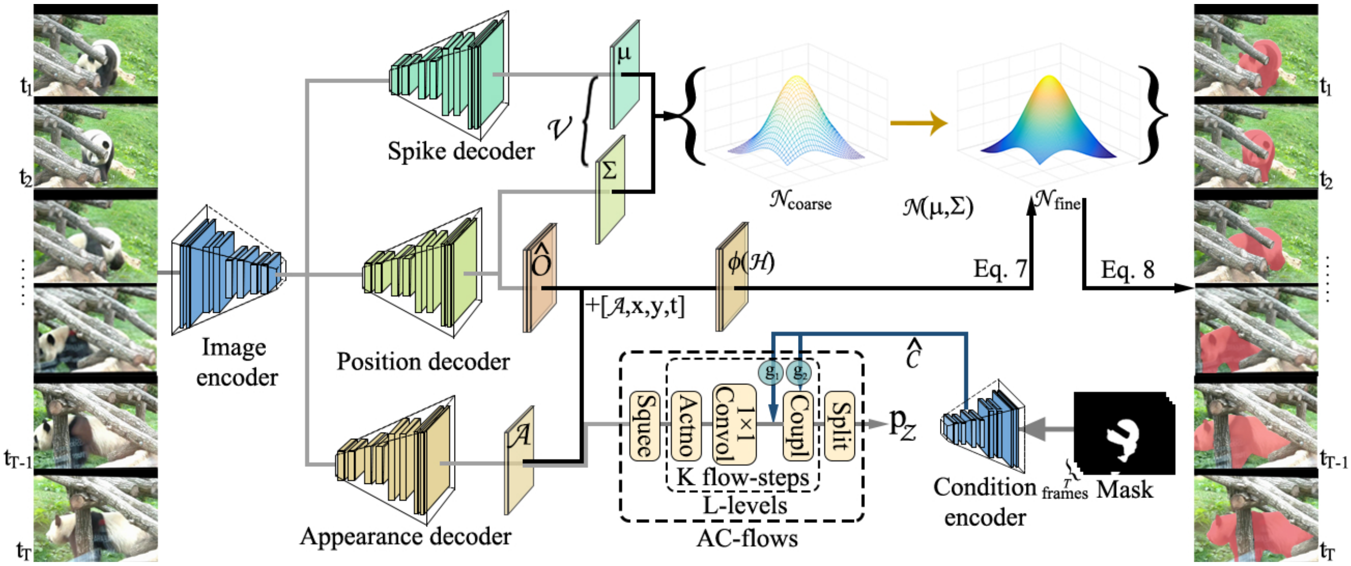

Figure 2: Overall architecture of our HEVis for video instance segmentation in the training phase. HEVis takes a video clip with T frames and its instance-level mask as input and outputs clip-level instance segmentation results. Based on features from the image encoder, three decoders produce the mean μ and covariance matrix Σ of a multi-variate Gaussian $\mathcal{N}(\mu, \Sigma)$, the offset vectors $\hat{O}$ and appearance embedding $\mathcal{A}$. μ and Σ are associated with position information $\mathcal{V}$ (*i.e.*, [x,y,t]) and are used to define the multi-variate Gaussian $\mathcal{N}(\mu, \Sigma)_{coarse}$ for each instance. The position decoder works as an intermediate that connects the other two decoders densely. $\hat{O}$ and the hierarchical embedding $[\mathcal{A}$, x, y, t] are fed to a novel mixing function (*i.e.*, ϕ) and are processed through Eq. 7, Eq. 8 to obtain the instance output tubes. Following the appearance decoder, AC-flows refines $\mathcal{N}(\mu, \Sigma)_{coarse}$ into $\mathcal{N}(\mu, \Sigma)_{fine}$ (*i.e.*, →) by optimizing $\mathcal{A}$. During training, instance masks are encoded and injected into AC-flows (*i.e.*, →) as a condition.

work is the first to exploit a flow architecture for VIS that provides superior results compared to state-of-the-art methods. Second, to effectively capture underlying appearance variance, we develop a factorized conditional flow-based embedding learning protocol.

3 METHOD

In this section, we propose a hierarchical variable Bayesian scheme for video instance segmentation: HEVis. Theoretically, we tackle VIS as a hierarchical coarse-to-fine iterative inference problem. As shown in Fig. 2, a coarse segmentation mask is obtained according to the position and shape of the instance, and then the details are inferred based on normalizing flows.

In §3.1, we first introduce a few preliminaries. In §3.2, we introduce the method to learn hierarchical feature distributions. §3.3 describes our network structure while §3.4 introduces the implementation details.

3.1 Preliminaries

Normalizing Flows: Normalizing flows [35] describe the transformation of a probability density through a sequence of *invertible mappings*. Given an observed datum variable $x \in \mathcal{X} \subseteq \mathbb{R}^d$ with an unknown real distribution $x \sim p_{\mathcal{X}}(x)$ and a simple distribution $p_{\mathcal{Z}}$ on a latent variable $z \in \mathcal{Z} \subseteq \mathbb{R}^d$, the *invertible mapping* defines an invertible function $f_\theta : \mathcal{X} \to \mathcal{Z}$ and applies it to the prior distribution:

$$p_{\mathcal{X}}(x) = p_{\mathcal{Z}}(f_\theta(x)) \left| \det \left(\frac{\partial f_\theta(x)}{\partial x^T} \right) \right|, \tag{1}$$

$$\log(p_{\mathcal{X}}(x)) = \log(p_{\mathcal{Z}}(f_\theta(x))) + \log \left(\left| \det \left(\frac{\partial f_\theta(x)}{\partial x^T} \right) \right| \right), \tag{2}$$

where $\frac{\partial f_\theta(x)}{\partial x^T}$ is the Jacobian of f_θ at x. f_θ is usually composed of a sequence of transformations: $f_\theta = f_1 \circ f_2 \circ \cdots \circ f_L$ complies with the name of *flows*. Typically, $p_{\mathcal{Z}}$ has a tractable density, such as a spherical multivariate Gaussian distribution: $p_{\mathcal{Z}} = \mathcal{N}(z; 0, \mathbf{I})$.

Deep Framework with Coupling Layers: NICE [8] and realNVP [9] proposed a deep learning framework for flow-based generative models using the coupling layer. Given a D-dimensional input x and $d < D$, to split the input tensor into two halves along the channel dimension $[x_{1:d}; x_{d+1:D}]$ and follow an affine coupling operation, the output y of the coupling layer is:

$$\begin{aligned} &\begin{cases} y_{1:d} &= x_{1:d} \\ y_{d+1:D} &= x_{d+1:D} \odot exp(s(x_{1:d})) + t(x_{1:d}) \end{cases} \\ \Longleftrightarrow &\begin{cases} x_{1:d} &= y_{1:d} \\ x_{d+1:D} &= (y_{d+1:D} - t(y_{1:d})) \odot exp(-s(y_{1:d})), \end{cases} \end{aligned} \tag{3}$$

where s and t represent scale and translation, $\odot$ is the Hadamard product or element-wise product. In this way, the determinant of a triangular matrix can be efficiently computed as the product of its diagonal terms.

3.2 Hierarchical Embedding Learning for Video Instance Learning

Before clarifying our model, we firstly introduce some basic notations and operations used in this paper. The main symbols used in this paper are summarized in Table 1.

Table 1: Symbol description.

Symbols	Descriptions
$\mathcal{X}$; C	input video clip and the conditional input
Y_{clip} ; G	the predicted instance segmentation mask tube and ground truth
$\mathcal{N}(\mu, \Sigma)$	a multivariate Gaussian with mean μ and covariance matrix Σ
$p_{\hat{e}}$	the probability that the spatio-temporal $\hat{e}$ belongs to the n^{th} instance
$\mathcal{E}$; $\mathcal{A}$; $\mathcal{H}$	the spatio-temporal embedding; the underlying appearance embedding; and the hierarchical embedding
$\mathcal{S}$	the spike probability map (i.e., μ)
O ; $\hat{O}$	the offset
$\phi(\cdot)$; $\psi(\cdot)$	the mixing function and the update function

In the context of VIS, the proposed HEVis aims to infer the instance mask in a coarse-to-fine manner. We first generate a coarse mask by modeling the motion and shape of all the instances as a 3D spatio-temporal volume. Then, we utilize density estimation through factorized-based conditional generative flows to learn fine-grained appearance embedding. Thus, our model can capture fine-grained appearance masks for each instance.

Coarse Embedding Learning: We handle VIS at a video clip level rather than at the frame level to preserve more spatio-temporal information. Specifically, given an input video clip $\mathcal{X} \in \mathbb{R}^{T \times 3 \times H \times W}$ with T frames, width W, height H, VIS aims to annotate each pixel in the video clip as from the background or one of N instances. This procedure returns the posterior class probabilities for each pixel location. Formally, let the set of the spatio-temporal feature embedding extracted from the video clip be $\{\mathbf{x}_i\}_i$. The feature embedding $\mathbf{x}_i$ at each position i is a $\hat{D}$-dimensional vector, and we model these feature embeddings with the distributions:

$$p\left(\mathbf{e}_i\right) = \sum\nolimits_{n=1}^{N} p(z_i = n) p(\mathbf{x}_i | z_i = n). \tag{4}$$

Each class-conditional density is a multi-variate Gaussian with mean μ and covariance matrix Σ:

$$p(\mathbf{x}_i | z_i = n) = \mathcal{N}(x_i | \mu, \Sigma). \tag{5}$$

The discrete random variable z_i in Eq. 4 assigns the observation x_i to a specific component $z_i = n$. We use a uniform prior $p(z_i = n) = 1/N$ for this variable, where N is the number of instances.

For the distribution of the embedding x_i, we can compute the mean and covariance as:

$$\begin{aligned} \mu &= \frac{1}{\hat{D}} \sum_j x_i^j \\ \Sigma &= \frac{1}{\hat{D}} \mathrm{diag} \sum_j (g^j - \mu)^2. \end{aligned} \tag{6}$$

For the spatio-temporal embedding $\hat{e} \in e_n$, $\forall n \subseteq \{1, ..., N\}$, $\hat{e} \subseteq \mathbb{R}^3$, we can use the probability density function of $\mathcal{N}(\mu_n, \Sigma_n)$ to calculate the probability that it belongs to the n^{th} instance:

$$p_{\hat{e}} = \frac{1}{(2\pi)^{\frac{\hat{D}}{2}} |\Sigma_n|^{\frac{1}{2}}} \exp\left(-\frac{1}{2}(\hat{e} - \mu_n)^T \Sigma_n^{-1} (\hat{e} - \mu_n)\right), \tag{7}$$

A high probability implies that the position embedding e_i tends to be the instance n, while a low probability implies that the embedding is more likely to be the background (or another instance). According to Eq. 7, we can obtain the predicted instance segmentation mask tube Y_{clip} by setting the probability threshold ω.

$$Y_{clip} = \begin{cases} \max p_{\hat{e}}, & \text{if} \, \exists \, p_{\hat{e}} > \omega \\ \text{background}, & \text{if} \, \forall \, p_{\hat{e}} > \omega, \end{cases} \tag{8}$$

where $p_{\hat{e}} \subseteq \mathbb{R}^N$.

For more complete representation of spatio-temporal information and to ensure good performance, the spatio-temporal embedding can be augmented as:

$$\phi(\mathcal{E}) = O + \mathcal{V}, \tag{9}$$

where $\phi(\cdot)$ denotes the mixing function, $\mathcal{V}$ is the position coordinate (*i.e.*, $[x, y, t] \subseteq \mathbb{R}^{T \times 3 \times H \times W}$) and $O \subseteq \mathbb{R}^{T \times 3 \times H \times W}$ is offsets to the $\mathcal{V}$ value of their respective locations. We use an extra network to learn the offset vectors. It is postulated that the reason for the good results obtained from this formulation is that the position vectors $\mathcal{V}$ already serves as a good initial embedding for instance separation; the network can then enhance this representation by producing offsets which embedding improves the segmentation behavior. Hence, the mixing function contains coarse contour information about each instance.

Refine Learning: In this method, based on position information, we learn the hierarchical embedding $\mathcal{H}$ to refine the prediction. As the segmentation targets undergo appearance variation (*i.e.*, fast motion, occlusion) in the video, it is meaningful to estimate the underlying appearance embedding $\mathcal{A}$ to handle these challenges. To this end, we define a novel mixing function for refining the inference status as follows:

$$\phi(\mathcal{H}) = \hat{O} + [\mathcal{V}, \psi(\mathcal{A})], \tag{10}$$

where $\hat{O} \subseteq \mathbb{R}^{T \times (3+D^A) \times H \times W}$, $\mathcal{A} \subseteq \mathbb{R}^{T \times D^A \times H \times W}$, [,] denotes the channel-wise concatenation and $\psi(\cdot)$ is a update function.

By addressing appearance changes as a density estimation issue, we introduce appearance-based conditional flows (AC-flows) to implement $\psi(\cdot)$. To recap, we utilize $\mathcal{A}$ to represent the appearance distribution among the temporal dimension, and $\psi(\cdot)$ learns to update appearance embedding $\mathcal{A}$.

The probability density of the appearance can be computed as:

$$p_{\mathcal{A}}(a; c, \theta) = p_{\mathcal{Z}}(f_\theta(a; c)) \left| \det\left(\frac{\partial f_\theta(a; c)}{\partial a^T}\right) \right|, \tag{11}$$

Given a simple known distribution $p_{\mathcal{Z}}$, AC-flows projects the appearance embedding $a \in \mathcal{A}$, $a \subseteq \mathbb{R}^{1 \times D^A \times H \times W}$ to a distribution, dependent on both the network parameters θ of ϕ and the conditioning input $c \in C$, through the change-of-variables formula. We use the instance mask to instantiate c for the sake of enforcing the appearance embedding containing more information about instance details. Moreover, we employ a mask encoder h to transform the conditional input c into intermediate representation $\hat{c} = h(c) \in \hat{C}$, and replace c in Eq. 11 with $\hat{c}$.

Additionally, to inject more direct spatio-temporal information transfers from the $\hat{c}$ into the AC-flows, we perform the following two operations (see Fig. 2) in the flow-step of the conditional affine coupling layer [27]:

First, we use a simple normalization function $g_1(\cdot)$ to predict an element-wise scaling and bias factor of $\hat{c}$ before the coupling transformation (Eq. 3):

$$\begin{cases} \hat{a} = exp(g_{1,s}(\hat{c})) \cdot a + g_{1,b}(\hat{c}) \\ a = exp(-g_{1,s}(\hat{c})) \cdot (\hat{a} - g_{1,b}(\hat{c})). \end{cases} \tag{12}$$

Then we apply another normalization function $g_2(\cdot)$ during the coupling transformation:

$$\begin{cases} \hat{y}_{1:d} = \hat{a}_{1:d} \\ \hat{y}_{d+1:D^A} = exp(g_{2,s}(\hat{a}_{d+1:D^A})) \cdot \hat{a}_{d+1:D^A} + g_{2,b}(\hat{a}_{d+1:D^A}), \end{cases} \tag{13}$$

where $\hat{a} = [\hat{a}_{1:d};\ \hat{a}_{d+1:D^A}]$ is a partition in the channel dimension. We implement normalization functions $g_1(\cdot)$ and $g_2(\cdot)$ with 1×1 convolutional layer with shared parameters. These two operations mentioned above are integrated into all K flow-steps with L levels.

Hence, AC-flows learn a more elaborated embedding to supplement the coarse multi-variate Gaussian modeling in the coarse embedding learning module. Overall, our network preserves spatial details and simultaneously models the appearance information.

Loss Function: To guide the network to capture hierarchical embedding, we design the following loss function. Given the video instance segmentation results $Y_{clip} \subseteq \mathbb{R}^{D^Y}$ where $D^Y = T \times 1 \times H \times W$ (Eq. 8), spike probability map $\mathcal{S}$ (*i.e.*, μ), and corresponding Gaussian covariance matrix Σ, the multi-variate Gaussian modeling can be posed as the minimization of the following loss:

$$\begin{aligned} L_{spike} = \frac{1}{D^Y} &\sum_{y \in Y_{clip},\, s \in \mathcal{S}} [y \ln s + (1-y) \ln(1-s)] \\ &+ \alpha \frac{1}{D^Y} \sum_{\sigma \in \Sigma} ||\sigma - \text{mean}(\Sigma)||^2, \end{aligned} \tag{14}$$

where the coefficient α is empirically set as 10. In Eq. 14, the first term is a binary cross-entropy (BCE) loss that is used to compute the spike distance (*i.e.*, the mean μ) of the Gaussian model. The second term is the L_2 regression loss to constrain the covariance. In this manner, the background pixels are regressed to zero while the foreground pixels are regressed to the Gaussian distributions for the object instances.

Furthermore, to learn hierarchical embedding $\mathcal{H}$, we apply a Lováse hings loss [49] over the prediction Y_{clip} and ground truth G for maximizing the intersection-over-union (IOU):

$$L_{hiera} = \arg\max \sum_{n=1}^{N} \frac{y_n \bigcap g_n}{y_n \bigcup g_n}, \text{ where } y_n \in Y_{clip},\ g_n \in G. \tag{15}$$

Finally, to guide the proposed AC-flows to learn to estimate the appearance variance, we tailor Eq. 2 and design the following conditional log-likelihood maximization loss function:

$$L_{appea} = \log(p_{\mathcal{Z}}(f_\theta(\mathcal{A};\hat{C}))) + \log\left(\left|\det\left(\frac{\partial f_\theta(\mathcal{A};\hat{C})}{\partial a^T}\right)\right|\right), \tag{16}$$

where $\mathcal{A}$ is the appearance embedding and $\hat{C}$ is the condition variance. In this way, the coarse embedding learning module can be trained jointly with the AC-flows by propagating gradients from the maximum likelihood loss through appearance embedding $\mathcal{A}$.

Considering Eqs. 14,15 and 16, the overall loss is defined as:

$$L_{total} = \beta_1 L_{spike} + \beta_2 L_{hiera} + \beta_3 L_{appea}, \tag{17}$$

For comprehensive consideration of different quantification factors, we follow the parameter setting in previous work [2], and empirically choose $\beta_1 = \beta_2 = \beta_3 = 1$ for a fair comparison.

3.3 HEVis Architecture

The overall HEVis architecture, depicted in Fig. 2, consists of two encoders (image encoder and condition encoder), three decoders, and the proposed AC-flows.

Encoder-Decoders: Both the image encoder and condition encoder have the same network architecture, except for inputs. These encoders adopt a 3D feature pyramid network (FPN) structure and output multi-scale feature maps. For decoders, the first one is the center decoder which regresses the instance spike. It augments the vanilla decoder with a sigmoid activation function, which is used to output a spike probability map $\mathcal{S}$ (*i.e.*, μ) containing the position and value of the expected value μ of the Gaussian distribution. Then, the position decoder outputs both the offset $\hat{O}$ and the covariance matrix Σ by two simple parallel 3D convolution layers after the vanilla decoder, respectively. Finally, the appearance decoder outputs appearance embedding $\mathcal{A}$ after the vanilla decoder and is followed by the appearance conditional flows (AC-flows) to learn the appearance distribution. Specifically, we discard the AC-flows module in the inference phase, i.e., our HEVis eliminates the AC-flow module and only takes the RGB-frames as the inputs during the network inference.

AC-flows: As the core component for appearance variance estimation, AC-flows is implemented by a multiple-scale architecture [17] with L-level, and each level contains K number of flow-steps (see Fig. 2).

Each flow-step (Eqs. 12 and 13) in AC-flows consists of five different layers, firstly the squeeze layer performs a squeeze operation that effectively halves the spatial resolution. Then, the actnorm layer provides a channel-wise normalization, followed by an invertible 1×1 convolution layer. Next, we use a conditional affine coupling layer to enhance spatio-temporal information transmission. Finally, half of the channels are transferred to the next level in the split layer. Except conditional affine coupling layer, we keep the rest layers in its standard un-conditional form as [17].

3.4 Implementation Details

Training: For network training, we set the input size of the video clip as $T = 8$, $H = 352$, $W = 640$ to maintain high computational efficiency. ResNet-50 and ResNet-101 [15] serve as the backbone architecture, which are pre-trained using the weights trained on the image instance segmentation dataset: the microsoft common objects in context (COCO) [20]. Following the standard training protocol in [2], we use on-the-fly random affine transformations and motion blur to augment video data on Pascal VOC [11] and COCO. For COCO, we use object classes that overlap with the YouTube-VIS$_{19}$. The probability threshold ω in Eq. 8 is set to 0.5.

Table 2: State-of-the-art comparison on the YouTube-VIS$_{19}$.

	Method	Backbone	FPS	AP	AP_{50}	AP_{75}	AR_1	AR_{10}
Multiple-stage	OSMN MaskProp [48]	ResNet-50	-	23.4	36.5	25.7	28.9	31.1
	FEELVOS [37]	ResNet-50	-	26.9	42.0	29.7	29.9	33.4
	IoUTracker+ [47]	ResNet-50	-	23.6	39.2	25.5	26.2	30.9
	OSMN [48]	ResNet-50	-	27.5	45.1	29.1	28.6	33.1
	DeepSORT [30]	ResNet-50	-	26.1	42.9	26.1	27.8	31.3
	MaskTrack R-CNN [47]	ResNet-50	32.0	30.3	51.1	32.6	31.0	35.5
	MaskTrack R-CNN [47]	ResNet-101	20.0	31.8	53.0	33.6	33.2	37.6
	SeqTracker [47]	ResNet-50	-	27.5	45.7	28.7	29.7	32.5
Single-stage	STEm-Seg [2]	ResNet-50	10.5	30.6	50.7	33.5	31.6	37.0
	STEm-Seg [2]	ResNet-101	10.0	34.6	55.8	37.9	34.4	41.6
	HEVis	ResNet-50	13.0	32.7	53.5	33.6	32.9	38.2
	HEVis	ResNet-101	12.0	35.3	53.5	34.6	34.9	40.2

For AC-flows, the spatial dimensions of the input are 1/4 of the input video clip. We set $K = 32$, $L = 4$ in the flow-step. The whole training procedure of our HEVis consists of two steps. We first train AC-flows for 2k iterations by freezing the rest model parameters. Then, we train the entire model end-to-end in 24k iterations.

We implemented the proposed approach using PyTorch [31]. The model is trained on a single RTX 3090 GPU of 24 GB RAM. The batch size is set to 32. We optimize the loss function using the standard SGD solver, with a momentum of 0.9 and a weight decay of 0.0001. To schedule the learning rate, we employ the exponential decay scheduler, where the base learning rate is 5×10^{-4} with the decay factor as 0.01.

Considering that VIS needs to predict category labels for each instance, we add an extra semantic decoder based on the vanilla decoder after the image encoder. It performs category prediction for all pixels in the input clip and is trained using a standard cross-entropy loss.

Inference: After training, we apply our learned model to the unseen videos directly. We process each testing video in a sequential manner and perform temporal sampling [38] to build a video clip. Each clip is modeled as a 3D spatio-temporal volume and processed in a temporally sequential manner. Starting from the first T frames, our network first generates a set of instance variables: the spike probability map $\mathcal{S}$, the covariance matrix Σ and the hierarchical embedding $\mathcal{H}$. The entire inference procedure is as follows:

(1) Find the spike $s_n = \arg\max \mathcal{S}$, $\forall n \in \{1, ..., N\}$;
(2) Build the Gaussian distribution $\mathcal{N}(s_n, \Sigma_n)$;
(3) Compute the probability of the corresponding hierarchical embedding $h \in \mathcal{H}$ of each pixel's belonging to the instance n by Eq. 7;
(4) Generate the predicted instance segmentation mask $y_n \in Y_{clip}$ by Eq. 8;
(5) Delete elements in $\mathcal{S}$, Σ, $\mathcal{H}$ that have already been allocated;
(6) Repeat steps (1)-(5) until all elements in $\mathcal{S}$ have been allocated, or the next spike in $\mathcal{S}$ falls below 0.5.

4 EXPERIMENTS

We first report the performance on the main task: video instance segmentation (§4.1). Then, in §4.2, to further demonstrate the advantages of our model, we test it on an additional task: unsupervised video object segmentation (UVOS). Finally, we conduct an ablation study in §4.3.

4.1 Main Task: Video Instance Segmentation

4.1.1 Experimental Setup. To demonstrate the effectiveness of our method, we apply it to the famous VIS dataset: **YouTube-VIS$_{19}$** [19]. YouTube-VIS$_{19}$ dataset contains 2,883 high-quality YouTube videos with 131k object instances spanning 40 known categories. The performance is measured in terms of the average precision (*AP*) and average recall (*AR*) metrics.

4.1.2 Quantitative Performance . We compare HEVis against some state-of-the-art VIS methods in Table 2. Overall, our model outperforms all the contemporary methods and sets a new state-of-the-art in terms of *AP* (32.7%) on YouTube-VIS$_{19}$ *val* set. Notably, our single-stage, proposal-free method obtains a significantly higher score compared to representative multiple-stage methods (*i.e.*, OSMN [48], FEELVOS [37], SeqTracker [47], DeepSORT [30] and MaskTrack R-CNN [47]. Specifically, the efficiency of HEVis is greater than that of the state-of-the-art MaskTrack R-CNN [47] by 2.4% and 3.3% in *AP* using ResNet-50 and ResNet-101 backbone, respectively. Meanwhile, in terms of AP, our method outperforms the one-stage competitor, STEm-Seg [2] by a large margin (1.9% and 0.7%), respectively. Considering that both methods have the advantage of the same training protocol, the performance gain mainly comes from the proposed hierarchical embedding learning scheme. Furthermore, we report the segmentation speed by averaging the inferred time for all instances. It can be observed that our model maintains favorable inference speed while maintaining optimal performance.

4.1.3 Qualitative Results. We further present the qualitative comparison results of YouTubeVIS in Fig. 3. For further results we refer to the supplementary material. By benefitting from the learning distribution of the appearance, our method can handle various challenging scenarios well compared to STEm-Seg [2], even in cases of the severe occlusion. On these challenging videos, coarse learning can serve as a good initial location for instance separation. Then, refine learning can enhance the distinguishing representation of the instance by the appearance distribution. Overall, both quantitative and qualitative results verify the effectiveness of the proposed hierarchical embedding-based VIS approach.

Table 3: Results on the DAVIS$_{19}$ val and test sets for the UVOS.

Dataset		DAVIS$_{19}$ *val*						
Method		RVOS [36]	VSD [47]	AGS [43]	KIS [7]	AGNN [39]	STEm-Seg [2]	**HEVis**
$\mathcal{J}$	Mean ↑	36.8	51.7	55.5	-	58.9	61.5	64.8
	Recall ↑	40.2	-	61.6	-	65.7	70.4	74.3
	Decay ↓	0.5	-	7.0	-	11.7	-4	-5
$\mathcal{F}$	Mean ↑	45.7	61.4	59.5	-	63.2	67.8	68.2
	Recall ↑	46.4	-	62.8	-	67.1	75.5	77.5
	Decay ↓	1.7	-	9.0	-	11.7	1.2	1.0
$\mathcal{J}$&$\mathcal{F}$	Mean ↑	43.7	56.6	57.5	59.9	61.1	64.7	66.5
Dataset		DAVIS$_{19}$ *test*						
Method		RVOS [36]	VSD [47]	AGS [43]	KIS [7]	AGNN [39]	STEm-Seg [2]	**HEVis**
$\mathcal{J}$	Mean ↑	17.7	51.7	42.1	50.0	43.1	19.4	50.3
	Recall ↑	16.2	59.9	48.5	58.9	49.0	15.1	59.2
	Decay ↓	1.6	21.7	2.6	8.4	-1.4	-2.3	-3.1
$\mathcal{F}$	Mean ↑	27.3	61.4	49.0	58.3	51.5	22.4	58.1
	Recall ↑	24.8	65.7	51.5	62.1	54.5	13.9	66.1
	Decay ↓	1.8	15.7	2.6	11.4	2.2	-2.4	-3.4
$\mathcal{J}$&$\mathcal{F}$	Mean ↑	22.5	56.5	45.6	54.2	47.3	20.9	54.2

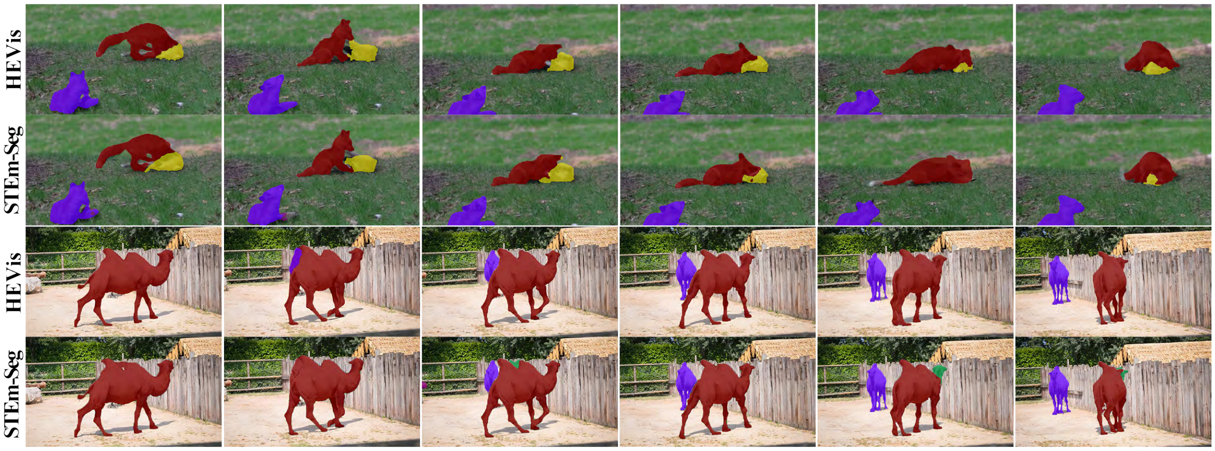

Figure 3: Qualitative results and comparison with STEm-Seg [2] on the YouTube-VIS$_{19}$ validation dataset [19] (row 1 and 2) and the DAVIS$_{19}$ dataset [5] (row 3 and 4).

4.2 Additional Task: Unsupervised Video Object Segmentation

4.2.1 Experimental Setup. We perform experiments on the recent challenging UVOS dataset: **DAVIS$_{19}$** [4][5][32][33]. DAVIS$_{19}$ consists of 90 videos (60 for training and 30 for validation) that contain multiple moving instances. The evaluation metric is $\mathcal{J}$&$\mathcal{F}$, the mean value of $\mathcal{J}$-score and $\mathcal{F}$-score. $\mathcal{J}$-score is the average of the IoU between the predicted and ground truth mask tubes, and $\mathcal{F}$-score is the accuracy of the predicted mask boundaries against the ground truth. Our HEVis model can be viewed as a framework for refining position information using appearance information.

4.2.2 Quantitative Performance. We report the results of UVOS on the DAVIS$_{19}$ *val* and *test* sets. A performance comparison is presented in Table 3. Among the existing methods such as: AGS [43], RVOS [36], VSD [47], KIS [7], AGNN [39] and STEm-Seg [2], our simple setup outperforms all the methods compared in terms of mean $\mathcal{J}$&$\mathcal{F}$ (66.5%), mean $\mathcal{J}$ (64.8%) and mean $\mathcal{F}$ (68.2%) on *val* set. Notably, our method obtains a significantly higher score for both regional similarity and contour accuracy compared to several representative methods using heuristic post-processing that handle each instance independently, such as KIS [7]. Meanwhile, compared to AGS [43], RVOS [36], VSD [47], AGNN [39] and STEm-Seg [2], our method outperforms these methods by a large margin. For the speed measured by FPS (frames per second), HEVis shows a significant advantage among STEm-Seg [2], achieving 12.0 FPS with the ResNet-101 backbone. The major reason is that our method handles the video on the clip level instead of the frame level in MaskTrack R-CNN [47]. This makes the data loading process time-consuming in the algorithm implementation. However, our model can be accelerated by parallel processing.

For completeness, we also evaluate our approach using the DAVIS$_{19}$ *test* set. This subset is considerably more challenging than the *val* set because the heavy and long-term occlusions among instances belonging to the same category are more frequent. The performance of all the compared methods degrades on this dataset. However, our method obtains the second-best score. We attribute the performance advances to the proposed coarse-to-fine mechanism, which helps to obtain more accurate segmentation. our method achieves an overall performance of 54.20% while the second-best method KIS [7] is 54.15% (the average of 50.0% in $\mathcal{J}$ score, 58.3% in $\mathcal{F}$ score) accurately. Therefore, our method is slightly higher than KIS [7] actually.

4.2.3 Qualitative Results. The qualitative comparison result of DAVIS is given in Fig. 3 (see supplementary for more results). Specifically, all object instances undergo severe deformation and suffer from the occlusion. However, through hierarchical embedding learning, our network handles these challenges well.

4.3 Ablation Study

In this section, we analyze the effect of the individual components of our method on the final performance of YouTube-VIS$_{19}$ [19], using the ResNet-50 backbone. Table 4 shows the results of diagnostic experiments. For each version, we retrain the entire network from scratch using the same procedure.

Table 4: Ablation Study on YouTubeVIS.

Component	Module	*AP*	Δ*AP*
Baseline	-	19.6	-
Appearance Embedding ($a \in \mathbb{R}^{1\times D^A\times H\times W}$)	D^A=1	21.2	+1.6
	D^A=2	22.3	+2.7
	D^A=3	24.5	+4.9
	D^A=4	23.8	+4.2
	D^A=5	20.6	+1.0
AC-flows	*w.*	31.4	+11.8
Sampling strategy [dense, sparse, multi-scale]	[✓,×,×]	31.6	+12.0
	[×,✓,×]	32.2	+12.6
	[✓,×,✓]	32.4	+12.8
	[×,✓,✓]	32.7	+13.1

Baseline: Our baseline constitutes a version where the model is trained directly by coarse learning, which is not subsequently refined by appearance embedding. In other words, the baseline variant is defined as the model without the appearance decoder, AC-flows, and condition encoder. Then, we retrain the entire network from scratch using the same procedure. In this case, our HEVis degenerates into a multivariate Gaussian model for coarse location. Furthermore, we do not adopt any sampling strategy during inference.
Appearance Embedding: We first study the impact on appearance embedding learning. As observed in Table 4, removing appearance embedding leads to huge performance degradation (Δ*AP*: 4.9%). The results clearly demonstrate that the introduced appearance embedding learning is an essential component in our VIS approach. Furthermore, we study the dimensions a of the appearance embedding. Compared to the identity baseline that only uses position embedding, imparting a novel mixing function with the appearance embedding of $D^A = 1$ improves the *AP* from 19.6% to 21.2%. Therefore, we can observe that when D^A changes from 1 to 3, the quantitative results show increased performance with an enhanced appearance embedding dimension. When we further increase D^A, the final performance becomes worse when using the straightforward way to superimpose the dimensions of appearance embedding. The main reason lies in the balance of the dimensions between appearance embedding and position embedding.

Table 5: The quantitative evaluation of condition embedding injection.

ℓ^{th}	(a)	(b)	(c)	(d)	(e)
ℓ=1		✓	✓	✓	✓
ℓ=2			✓	✓	✓
ℓ=3				✓	✓
ℓ=4					✓
AP	28.5	29.2	29.7	31.1	31.4

AC-flows: Next, we assess the importance of the proposed AC-flows according to the segmentation results based on $D^A = 3$. From the eighth row of Table 4, it is observed that AC-flows also help to improve the segmentation performance which is 31.4% *AP*. This suggests that a conditional probability density estimate enhances the appearance distribution boundary to facilitate instance-level discrimination.

We further investigate the influence of the conditional coupling layer. The results are shown in Table 5, where ℓ^{th} represents that tailoring the vanilla coupling layer into factorized conditional coupling layer. We can observe that greater performance gain is achieved by inserting more conditional coupling layers.
Sampling strategy: Finally, we investigate the influence of the sampling strategy on the inference process. We integrate two sampling strategies into the model inference procedure: dense sampling among successive frames and sparse sampling across multiple frames randomly. We can see that sparse sampling brings more performance improvement (32.2% versus 31.6%) as this rule makes the input video clips have larger receptive field. So the proposed method can capture more temporal information. Moreover, we further boost our performance to 32.7% by applying a multi-scale strategy during inference (32.7% versus 32.4%).

5 CONCLUSION

In this paper, we have proposed an instance video segmentation method, HEVis, from a hierarchical embedding learning view. We started with the observation that most current VIS methods fall into the segmentation-by-detection paradigm with several independent modules. Based on this insight, we proposed to learn a holistic generative model for capturing the spatio-temporal feature embedding and appearance feature embedding in an end-to-end way. Correspondingly, we leveraged a concise mixture model to represent spatio-temporal embeddings. Moreover, we exploited normalizing flows to estimate the underlying appearance embedding variance. These two embedding learning procedures are integrated into a hierarchical Bayesian learning framework. In this way, our model infers all the video instances in a single forward pass. The experiment shows that each component of our method is highly effective and achieves new state-of-the-art results on both the video instance segmentation and unsuperivsed video object segmentation datasets.In the future, we plan to tailor the flow model into an invertible version and extend our work to other video analysis tasks, such as video object detection [22].

6 ACKNOWLEDGMENTS

This work is partially supported by the National Natural Science Foun- dation of China (No. 61876098), the Taishan Scholar Project of Shandong Province (tsqn202103088), and the Fundamental Research Funds of Shandong University.

REFERENCES

[1] Lynton Ardizzone, Carsten Lüth, Jakob Kruse, C. Rother, and U. Köthe. 2019. Guided Image Generation with Conditional Invertible Neural Networks. *ArXiv* abs/1907.02392 (2019).

[2] Ali Athar, S. Mahadevan, Aljosa Osep, L. Leal-Taixé, and B. Leibe. 2020. STEm-Seg: Spatio-temporal Embeddings for Instance Segmentation in Videos. In *ECCV*.

[3] Gedas Bertasius and Lorenzo Torresani. 2020. Classifying, segmenting, and tracking object instances in video with mask propagation. In *CVPR*.

[4] Sergi Caelles, Alberto Montes, Kevis-Kokitsi Maninis, Yuhua Chen, Luc Van Gool, Federico Perazzi, and Jordi Pont-Tuset. 2018. The 2018 DAVIS Challenge on Video Object Segmentation. *arXiv:1803.00557* (2018).

[5] Sergi Caelles, Jordi Pont-Tuset, Federico Perazzi, Alberto Montes, Kevis-Kokitsi Maninis, and Luc Van Gool. 2019. The 2019 DAVIS Challenge on VOS: Unsupervised Multi-Object Segmentation. *arXiv:1905.00737* (2019).

[6] Jiale Cao, Rao Muhammad Anwer, Hisham Cholakkal, Fahad Shahbaz Khan, Yanwei Pang, and Ling Shao. 2020. Sipmask: Spatial information preservation for fast image and video instance segmentation. In *ECCV*.

[7] Donghyeon Cho, Sungeun Hong, Sungil Kang, and Jiwon Kim. 2019. Key Instance Selection for Unsupervised Video Object Segmentation. *arXiv:1906.07851* (2019).

[8] Laurent Dinh, David Krueger, and Yoshua Bengio. 2014. Nice: Non-linear independent components estimation. *arXiv preprint arXiv:1410.8516* (2014).

[9] Laurent Dinh, Jascha Sohl-Dickstein, and Samy Bengio. 2016. Density estimation using real nvp. *arXiv preprint arXiv:1605.08803* (2016).

[10] Minghui Dong, Jian Wang, Yuanyuan Huang, Dongdong Yu, Kai Su, Kaihui Zhou, Jie Shao, Shiping Wen, and Changhu Wang. 2019. Temporal Feature Augmented Network for Video Instance Segmentation. In *ICCV*.

[11] Mark Everingham, Luc Van Gool, Christopher K. I. Williams, John M. Winn, and Andrew Zisserman. 2010. The Pascal Visual Object Classes (VOC) Challenge. *Int. J. Comput. Vis.* 88, 2 (2010), 303–338.

[12] Qianyu Feng, Zongxin Yang, Peike Li, Yunchao Wei, and Yi Yang. 2019. Dual Embedding Learning for Video Instance Segmentation. In *ICCV Workshops*.

[13] Ian J. Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron C. Courville, and Yoshua Bengio. 2014. Generative Adversarial Networks. *arXiv:1406.2661* (2014).

[14] Kaiming He, Georgia Gkioxari, Piotr Dollár, and Ross B. Girshick. 2017. Mask R-CNN. In *ICCV*.

[15] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. 2016. Deep Residual Learning for Image Recognition. In *CVPR*.

[16] Dahun Kim, Sanghyun Woo, Joon-Young Lee, and In So Kweon. 2020. Video Panoptic Segmentation. In *CVPR*.

[17] Diederik P. Kingma and Prafulla Dhariwal. 2018. Glow: Generative Flow with Invertible 1x1 Convolutions. In *NeurIPS*.

[18] Diederik P. Kingma and Max Welling. 2014. Auto-Encoding Variational Bayes. (2014).

[19] Chung-Ching Lin, Ying Hung, Rogério Feris, and Linglin He. 2020. Video Instance Segmentation Tracking With a Modified VAE Architecture. In *CVPR*.

[20] Tsung-Yi Lin, Michael Maire, Serge J. Belongie, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollár, and C. Lawrence Zitnick. 2014. Microsoft COCO: Common Objects in Context. In *ECCV*.

[21] Xiaoyu Liu, Haibing Ren, and Tingmeng Ye. 2019. Spatio-Temporal Attention Network for Video Instance Segmentation. In *ICCV Workshops*.

[22] Xiankai Lu, Chao Ma, Jianbing Shen, Xiaokang Yang, Ian Reid, and Ming-Hsuan Yang. 2020. Deep Object Tracking with Shrinkage Loss. *IEEE Transactions on Pattern Analysis and Machine Intelligence* (2020), 1–1.

[23] Xiankai Lu, Wenguan Wang, Chao Ma, Jianbing Shen, Ling Shao, and Fatih Porikli. 2019. See More, Know More: Unsupervised Video Object Segmentation With Co-Attention Siamese Networks. In *CVPR*.

[24] Xiankai Lu, Wenguan Wang, Danelljan Martin, Tianfei Zhou, Jianbing Shen, and Van Gool Luc. 2020. Video Object Segmentation with Episodic Graph Memory Networks. In *ECCV*.

[25] Xiankai Lu, Wenguan Wang, Jianbing Shen, David Crandall, and Jiebo Luo. 2020. Zero-Shot Video Object Segmentation with Co-Attention Siamese Networks. *IEEE Transactions on Pattern Analysis and Machine Intelligence* (2020).

[26] Xiankai Lu, Wenguan Wang, Jianbing Shen, Yu-Wing Tai, David J Crandall, and Steven CH Hoi. 2020. Learning video object segmentation from unlabeled videos. In *CVPR*.

[27] Andreas Lugmayr, Martin Danelljan, Luc Van Gool, and Radu Timofte. 2020. SRFlow: Learning the Super-Resolution Space with Normalizing Flow. In *ECCV*.

[28] Jonathon Luiten, Philip H. S. Torr, and Bastian Leibe. 2019. Video Instance Segmentation 2019: A Winning Approach for Combined Detection, Segmentation, Classification and Tracking. In *ICCV Workshop*.

[29] Eslam Mohamed, Mahmoud Ewaisha, Mennatullah Siam, Hazem Rashed, Senthil Kumar Yogamani, and Ahmad El Sallab. 2020. InstanceMotSeg: Real-time Instance Motion Segmentation for Autonomous Driving. *arXiv:2008.07008* (2020).

[30] Daniel Munoz, Nicolas Vandapel, and Martial Hebert. 2009. Onboard contextual classification of 3-D point clouds with learned high-order Markov Random Fields. In *IEEE International Conference on Robotics and Automation, ICRA*. IEEE, 2009–2016.

[31] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Köpf, Edward Yang, Zachary DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala. 2019. PyTorch: An Imperative Style, High-Performance Deep Learning Library. (2019).

[32] Federico Perazzi, Jordi Pont-Tuset, Brian McWilliams, Luc Van Gool, Markus H. Gross, and Alexander Sorkine-Hornung. 2016. A Benchmark Dataset and Evaluation Methodology for Video Object Segmentation. In *CVPR*.

[33] Jordi Pont-Tuset, Federico Perazzi, Sergi Caelles, Pablo Arbelaez, Alexander Sorkine-Hornung, and Luc Van Gool. 2017. The 2017 DAVIS Challenge on Video Object Segmentation. *arXiv:1704.00675* (2017).

[34] Albert Pumarola, Stefan Popov, Francesc Moreno-Noguer, and Vittorio Ferrari. 2020. C-Flow: Conditional Generative Flow Models for Images and 3D Point Clouds. In *CVPR*.

[35] Danilo Jimenez Rezende and Shakir Mohamed. 2015. Variational Inference with Normalizing Flows. In *International conference on machine learning*, Vol. 37. PMLR, 1530–1538.

[36] Carles Ventura, Miriam Bellver, Andreu Girbau, Amaia Salvador, Ferran Marqués, and Xavier Giró-i-Nieto. 2019. RVOS: End-To-End Recurrent Network for Video Object Segmentation. In *CVPR*.

[37] Paul Voigtlaender, Yuning Chai, Florian Schroff, Hartwig Adam, Bastian Leibe, and Liang-Chieh Chen. 2019. FEELVOS: Fast End-To-End Embedding Learning for Video Object Segmentation. In *CVPR*.

[38] Limin Wang, Yuanjun Xiong, Zhe Wang, Yu Qiao, Dahua Lin, Xiaoou Tang, and Luc Van Gool. 2016. Temporal Segment Networks: Towards Good Practices for Deep Action Recognition. In *ECCV*.

[39] Wenguan Wang, Xiankai Lu, Jianbing Shen, David J. Crandall, and Ling Shao. 2019. Zero-Shot Video Object Segmentation via Attentive Graph Neural Networks. In *ICCV*.

[40] Wenguan Wang, Jianbing Shen, Xiankai Lu, Steven C. H. Hoi, and Haibin Ling. 2021. Paying attention to video object pattern understanding. 43, 7 (2021), 2413–2428.

[41] Wenguan Wang, Jianbing Shen, Fatih Porikli, and Ruigang Yang. 2018. Semi-supervised video object segmentation with super-trajectories. *IEEE TPAMI* 41, 4 (2018), 985–998.

[42] Wenguan Wang, Jianbing Shen, Ruigang Yang, and Fatih Porikli. 2017. Saliency-aware video object segmentation. *IEEE TPAMI* 40, 1 (2017), 20–33.

[43] Wenguan Wang, Hongmei Song, Shuyang Zhao, Jianbing Shen, Sanyuan Zhao, Steven C. H. Hoi, and Haibin Ling. 2019. Learning Unsupervised Video Object Segmentation Through Visual Attention. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 3064–3074.

[44] Wenguan Wang, Tianfei Zhou, Fatih Porikli, David Crandall, and Luc Van Gool. 2021. A Survey on Deep Learning Technique for Video Segmentation. arXiv:2107.01153

[45] Christina Winkler, Daniel E. Worrall, Emiel Hoogeboom, and Max Welling. 2019. Learning Likelihoods with Conditional Normalizing Flows. *arXiv:1912.00042* (2019).

[46] Guandao Yang, Xun Huang, Zekun Hao, Ming-Yu Liu, Serge J. Belongie, and Bharath Hariharan. 2019. PointFlow: 3D Point Cloud Generation With Continuous Normalizing Flows. In *ICCV*.

[47] Linjie Yang, Yuchen Fan, and Ning Xu. 2019. Video Instance Segmentation. In *ICCV*.

[48] Linjie Yang, Yanran Wang, Xuehan Xiong, Jianchao Yang, and Aggelos K. Katsaggelos. 2018. Efficient Video Object Segmentation via Network Modulation. In *CVPR*.

[49] Jiaqian Yu and Matthew B. Blaschko. 2015. Learning Submodular Losses with the Lovász Hinge. In *ICML*.

[50] Andrei Zanfir, Eduard Gabriel Bazavan, Hongyi Xu, William T. Freeman, Rahul Sukthankar, and Cristian Sminchisescu. 2020. Weakly Supervised 3D Human Pose and Shape Reconstruction with Normalizing Flows. In *ECCV*.