β-MuliVariational AutoEncoder (βMVAE) for Entangled Representation Learning in Video Frames

Fatemeh Nokabadi

23 Jul 2024

Content

- Introduction
- Single object tracking agent with comprehensive action set
- β-MuliVariational AutoEncoder (BMVAE and BMVUNet)
- Results
- Conclusion

Introduction: Motivation of using RL in object tracking

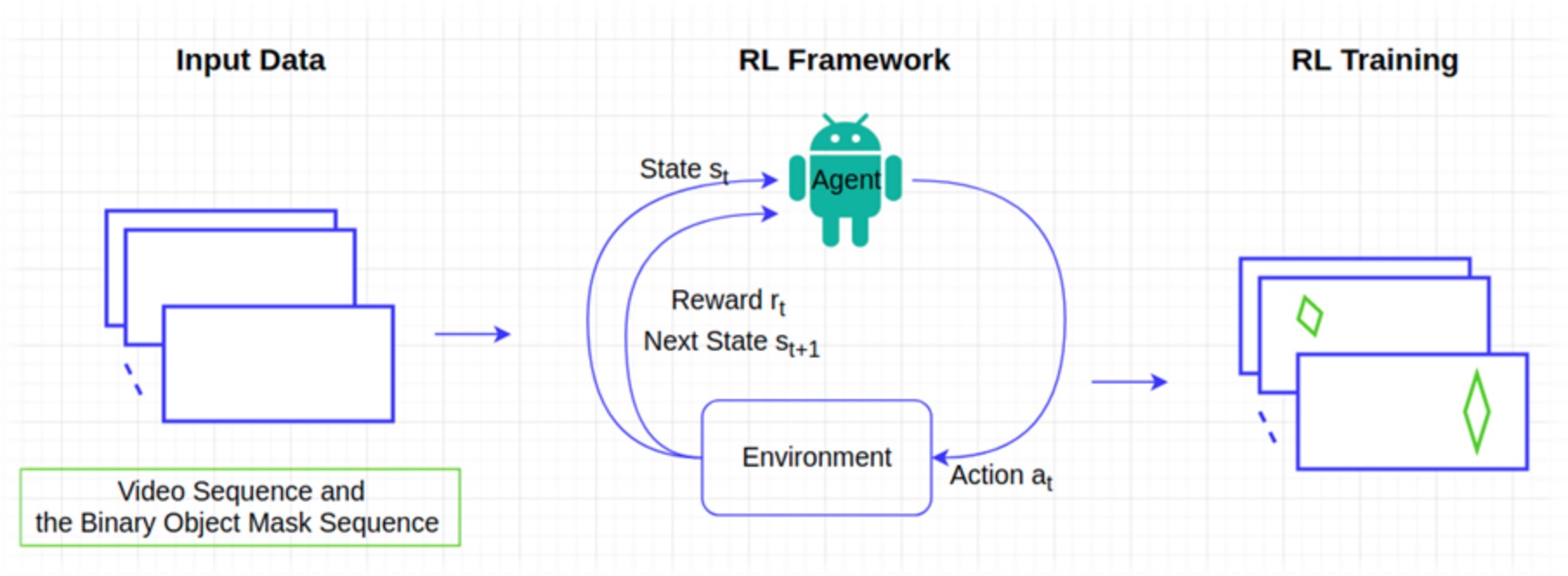
- Current trackers work based on the correlation of search and template regions

  - In the case that object is absent (occlusion) or deformed (appearance change), the target is lost!
  - Temporal information usage in tracker development is trivial!
  - Computation complexity of computing correlation for every step is high!

Suggestion: Using RL to learn the motion policy based on the target

Introduction

- Reinforcement learning to play episodic tasks



Introduction: Tracking as an episodic task

- Tracking dynamic as an episodic task:
  - Initial point: Initialize in the anchor frames with binary mask of the target
  - Repetitive task: Predict the object bbox starting from the second frame
  - Data charge**:** If we reach the end of video sequence, the state, prev bbox, and everything is charged with new video sequence.
  - Ending point: The tracking episode is ended whenever some principles are violated!

Episode violation rule

Whenever the predicted bbox is out of the current frame, the tracking episode must end!

Single Object Tracking Agent with Comprehensive Action Set

- What?

  Predict **rbbox** of target using RL agents

- Why?
  - Current trackers are basically using object recognition with slightly using temporal information
  - It learns motion policy conditioned on the frame and target. It may help trackers in case of occlusion and object deformation happens in videos

- How?

  Using Soft-Actor Critic algorithm with defining object tracking as an episodic task

Single Object Tracking Agent with Comprehensive Action Set

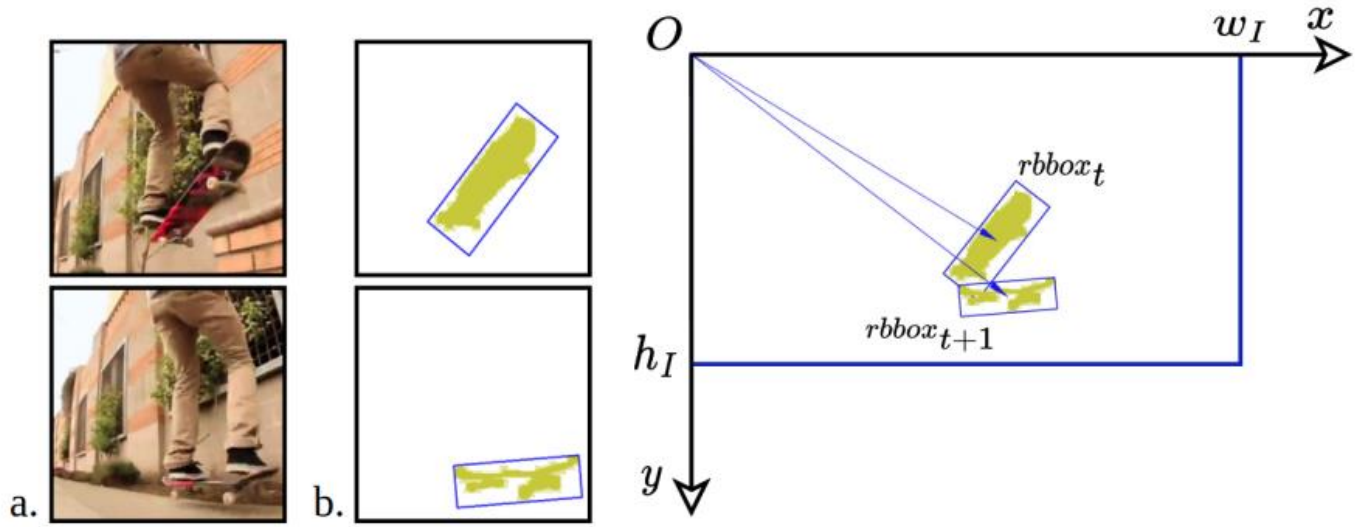


Figure 3: Let us consider two successive frames *rbbox*s. One can see that the *rbbox* at time step $t+1$ is the transformed *rbbox* at time step t, (a) (top) frame patch at time step t, and (bottom) frame patch at time step $t+1$, (b) (top) binary mask with *rbbox* at time step t, (bottom) binary mask with *rbbox* at time step $t+1$. On right, two *rbbox*s are shown in the frame as the foregrounds.

Single Object Tracking Agent with Comprehensive Action Set

4.1 Single moving object model in video sequence

In our method, the mask is used to compute the rotated bounding box ($rbbox$) in each time step. Then, we employ the ($rbbox$)s to describe the moving object characteristics in the frame as $rbbox = ([x, y], [w, h], \alpha)$ representing the center position, size and angle of the $rbbox$.

The underlying assumption of our method is that any motion of $rbbox$s between two successive frames is decipherable by three geometric transformations: translation, counterclockwise rotation, and scale. Figure 3 illustrates this idea for two consecutive frames. The object's position in the next frame $[x', y']$ can be calculated by transforming the current position $[x, y]$ as:

$$\begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} = G_t \times \begin{bmatrix} x \\ y \\ 1 \end{bmatrix} \tag{4}$$

where G_t is:

$$G_t = \begin{bmatrix} \Delta s_x \times cos\theta & -\Delta s_y \times sin\theta & \Delta x \\ \Delta s_x \times sin\theta & \Delta s_y \times cos\theta & \Delta y \\ 0 & 0 & 1 \end{bmatrix} \tag{5}$$

Single Object Tracking Agent with Comprehensive Action Set

1 Uniqueness of G_t in our motion model

It is essential to verify if it is feasible to calculate G_t for every two successive frames and whether it is a unique transformation in each time step t or not.

Let us assume the current and next $rbbox$ as:

$$rbbox_t = ([x, y], [s_x, s_y], \alpha) \tag{8}$$

$$rbbox_{t+1} = ([x', y'], [s'_x, s'_y], \alpha') \tag{9}$$

Following equations are used to calculate the scale in x axis, scale in y axis, rotation, translation in x axis, and the translation in y axis.

$$\Delta s_x = \frac{s'_x}{s_x} \tag{10}$$

$$\Delta s_y = \frac{s'_y}{s_y} \tag{11}$$

$$\theta = \alpha' - \alpha \tag{12}$$

$$t_x = x' - x \times \Delta s_x \times cos\theta + y \times \Delta s_y \times sin\theta \tag{13}$$

$$t_y = y' - y \times \Delta s_x \times sin\theta - y \times \Delta s_y \times cos\theta \tag{14}$$

Single Object Tracking Agent with Comprehensive Action Set

Therefore, there is one unique G_t for every two desirable $rbboxs$. To normalize the translation parameters, we divided the translations parameters by the image dimensions as follows

$$\Delta x = \frac{t_x}{w_I}, \Delta y = \frac{t_y}{h_I} \tag{15}$$

where w_I and h_I refer to the image's width and height. Also, we considered the $sin\theta$ for the rotation parameter to scale the numbers.

Single Object Tracking Agent with Comprehensive Action Set

- Initial results using SAC algorithm with our proposed environment:

CAST_rbbox, Classification + RL, Batch 29, Sequence 1

Rewards_1	IoU_1
[2.]	0.339252572028281
[2.]	0.1306305737288 21
[1.]	0.0370279668261 81
[0.]	0.0104380091261 69
[-2.]	0.0002306988919 25
[-4.]	0
[-6.]	0
[-8.]	0
[-10.]	0
[-12.]	0
[-14.]	0
[-16.]	0
[-18.]	0
[-20.]	0
[-22.]	0
[-24.]	0
[-26.]	0
[-24.]	1.54375434526364E-07
[-26.]	6.21071836810205E-08
[-28.]	2.99671475018 97E-08
[-30.]	1.33102541726376E-08
[-32.]	6.23544101840595E-09
[-34.]	2.59102861034391E-09
[-36.]	1.06204368815286E-09
[-38.]	4.92798583145904E-10
[-40.]	2.752504267413E-10
[-42.]	1.17068642017471E-10
[-44.]	4.86893729326096E-11
[-46.]	2.20113518658515E-11
[-48.]	8.74280749357008E-12
[-50.]	3.38856245933815E-12
[-52.]	1.25620313902235E-12
[-54.]	4.84293247797071E-13

CAST_rbbox, Segmentation + RL, Batch 29, Sequence 1

Rewards_1	IoU_1
[2.]	0.372594607622143
[2.]	0.146105865618557
[1.]	0.044882302941735
[0.]	0.006440263460028
[-2.]	1.42710322975926E-05
[-4.]	0
[-6.]	0
[-8.]	0
[-10.]	0
[-12.]	0
[-14.]	0
[-16.]	0
[-18.]	0
[-20.]	0
[-22.]	0
[-24.]	0
[-26.]	0
[-24.]	2.83212786213458E-07
[-26.]	2.16077152611097E-07
[-28.]	8.73082614295394E-08
[-30.]	3.879315911209E-08
[-32.]	1.65051589545186E-08
[-34.]	7.26465549838 01E-09
[-36.]	3.282537460552E-09
[-38.]	1.45787677769529E-09
[-40.]	6.05895136862915E-10
[-42.]	2.60447535090221E-10
[-44.]	1.05336314354659E-10
[-46.]	5.01161537765618E-11
[-48.]	2.12657189206497E-11
[-50.]	8.21340673038858E-12
[-52.]	3.0349086408604E-12
[-54.]	1.215898472637E-12

CAST_rbbox, Classification + RL, Batch 29, Sequence 3

Rewards_3	IoU_3
[2.]	0.414743297663495
[2.]	0.150601740845572
[2.]	0.06282799424146
[1.]	0.025500615456111
[0.]	0.011139163086075
[-1.]	0.005108955086161
[-3.]	0.002139453962015
[-5.]	0.000857870847461
[-7.]	0.000383733236489
[-9.]	0.000154295077522
[-11.]	6.34511530616465E-05
[-13.]	3.10000062687247E-05
[-15.]	1.3500369313903E-05
[-17.]	5.49759559964776E-06
[-19.]	2.20814084366608E-06
[-21.]	0
[-19.]	4.03201402876344E-07
[-21.]	1.72372724141369E-07
[-23.]	7.61563884141443E-08
[-25.]	3.1910872220167E-08
[-27.]	1.36680706235601E-08
[-29.]	5.71363398132521E-09
[-31.]	2.41896266670267E-09
[-33.]	1.06906445228598E-09
[-35.]	4.27503811670812E-10
[-37.]	1.70300902656891E-10
[-39.]	7.08292522519714E-11
[-41.]	2.80501348884246E-11
[-43.]	1.15675356919401E-11
[-45.]	4.80431200062415E-12
[-47.]	1.87944320202069E-12
[-49.]	9.41215217639529E-13
[-51.]	5.23250712763428E-13

CAST_rbbox, Segmentation + RL, Batch 29, Sequence 3

Rewards_3	IoU_3
[2.]	0.422184892924633
[2.]	0.215663340407351
[2.]	0.097398822256352
[1.]	0.039259451447375
[0.]	0.018422409253544
[-1.]	0.007957465076915
[-3.]	0.003197789072667
[-5.]	0.001327623813878
[-7.]	0.000542181734419
[-9.]	0.000239500620724
[-11.]	9.55458921787891E-05
[-13.]	4.28533600734462E-05
[-15.]	1.78305751118759E-05
[-17.]	8.08778891304118E-06
[-19.]	3.24642970038609E-06
[-21.]	1.31841694146932E-06
[-23.]	5.24842941290364E-07
[-25.]	2.13280825808869E-07
[-27.]	8.81735898612059E-08
[-29.]	3.72580700477645E-08
[-31.]	2.2862281731246E-08
[-33.]	9.20571935955614E-09
[-35.]	4.23949044631121E-09
[-37.]	1.83545596855464E-09
[-39.]	8.29846598283953E-10
[-41.]	3.8151736531271E-10
[-43.]	1.64794440986282E-10
[-45.]	6.6385587723691E-11
[-47.]	2.65162118034473E-11
[-49.]	1.16541476888618E-11
[-51.]	4.89065340657389E-12
[-53.]	2.16781767525248E-12
[-55.]	9.25465018032523E-13

Single Object Tracking Agent with Comprehensive Action Set

- Reason?
  - SAC is sampling action values from standard Gaussian distribution *N(0,I)*

- Problem?

  Are our motion parameters, i.e. actions, iid?

$$[\Delta s_x, \Delta s_y, \theta, \Delta x, \Delta y]$$

Data analysis? No, they are NOT iid!

Solution:

1- Compute the parameters using uniqueness of G_t
2- Assume MultiVariate Gaussian Distribution on data population
3- Learn this prior (MGD) from video frames

Single Object Tracking Agent with Comprehensive Action Set

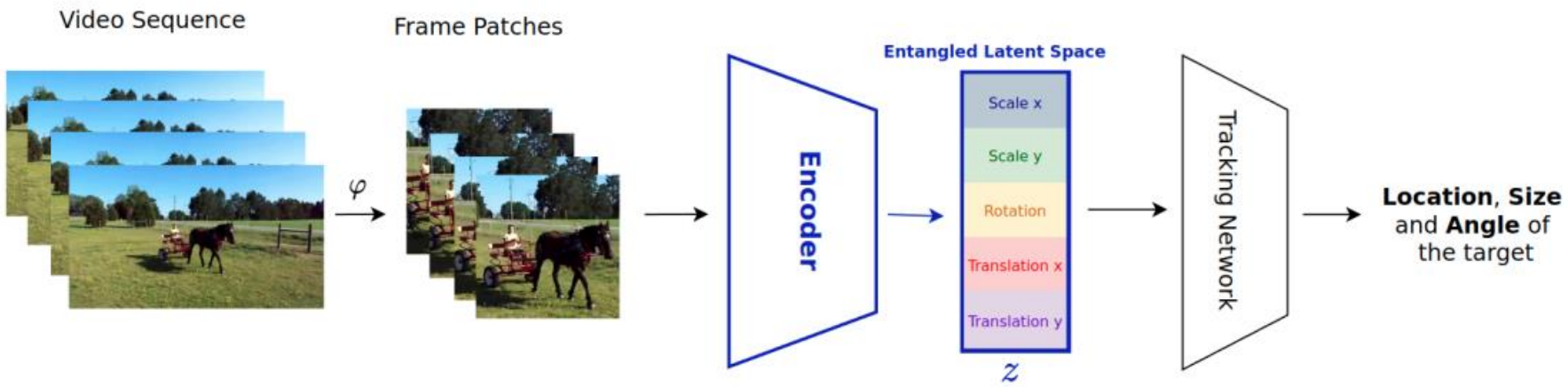


Figure 1: The general idea of our tracking method contains an encoder to place the variables into an appropriate multivariate Gaussian space where φ is the preprocessor to crop the video frames appropriately. This paper focuses on developing and training the encoder with the bottleneck . Once we have learned the action space, i.e. latent space, we use these multivariational inferences to form the MDP states and estimate the tracking parameters based on them.

β-MuliVariational AutoEncoder

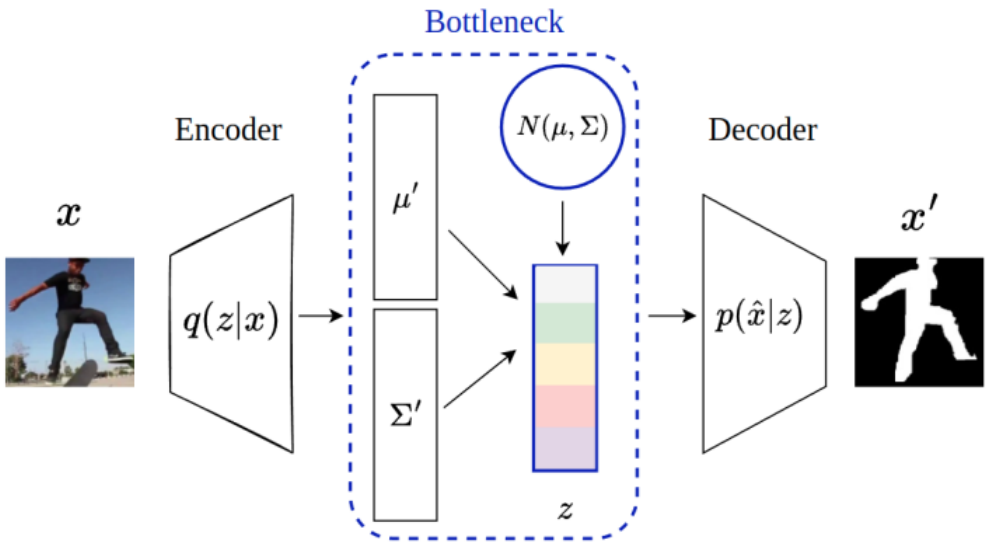


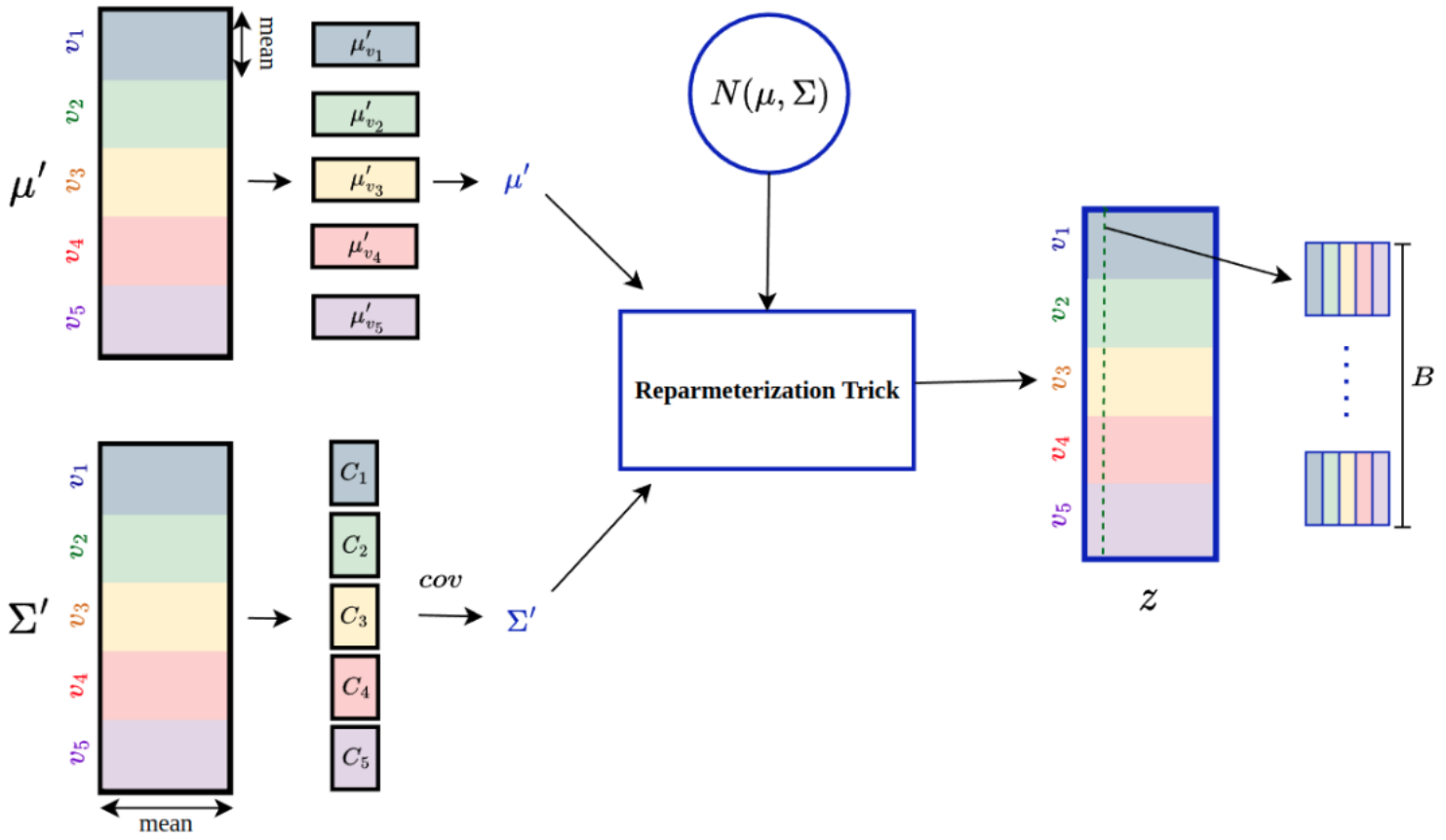
Figure 2: Overview of our proposed method, in which video frames are encoded into a sample set of the posterior, while the decoder is expected to estimate the object binary mask using the multivariational inferences.

$$\mathcal{L}_{\beta\text{VAE}} = -\mathbb{E}_{q_\phi(z|x)}\left[\log p_\theta(x|z)\right] + \beta D_{KL}\left(q_\phi(z|x)||p(z)\right)$$

$$\mathcal{L}_{\beta\text{MVAE}} = \mathcal{L}_{cons}(\hat{x}, gt) + \beta\mathcal{L}_{KL}$$

$$\mathcal{L}_{cons}(\hat{x}, gt) = \mathcal{L}_{ce}(\hat{x}, gt) + \mathcal{L}_{\mathcal{J}}(\hat{x}, gt)$$

β-MuliVariational AutoEncoder: Bottleneck



β-MuliVariational AutoEncoder: Rep. trick

Table 1: Lower Bound (LB) and Upper Bound (UB) of the latent distributions. These numbers are rounded floating points.

variables	$\mathcal{N}(\mu, \sigma)$	LB$(\mu - \sigma)$	UB$(\mu + \sigma)$
v_1	$\mathcal{N}(1.06, 0.52)$	0.55	1.59
v_2	$\mathcal{N}(1.06, 0.54)$	0.53	1.61
v_3	$\mathcal{N}(0, 0.43)$	-0.43	0.43
v_4	$\mathcal{N}(0.07, 0.34)$	-0.27	0.41
v_5	$\mathcal{N}(0.08, 0.74)$	-0.66	0.82

β-MuliVariational AutoEncoder: Rep. trick

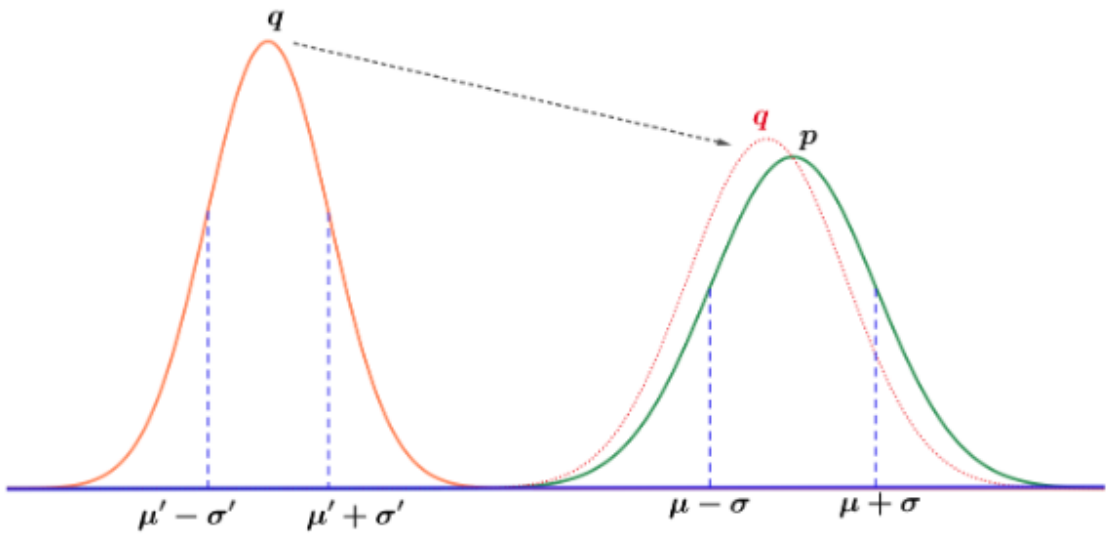


Figure 2.7: The reparameterization trick linearly maps the data distribution q to the target distribution p by considering two margins $\mu - \sigma$ and $\mu' - \sigma'$ for the data population.

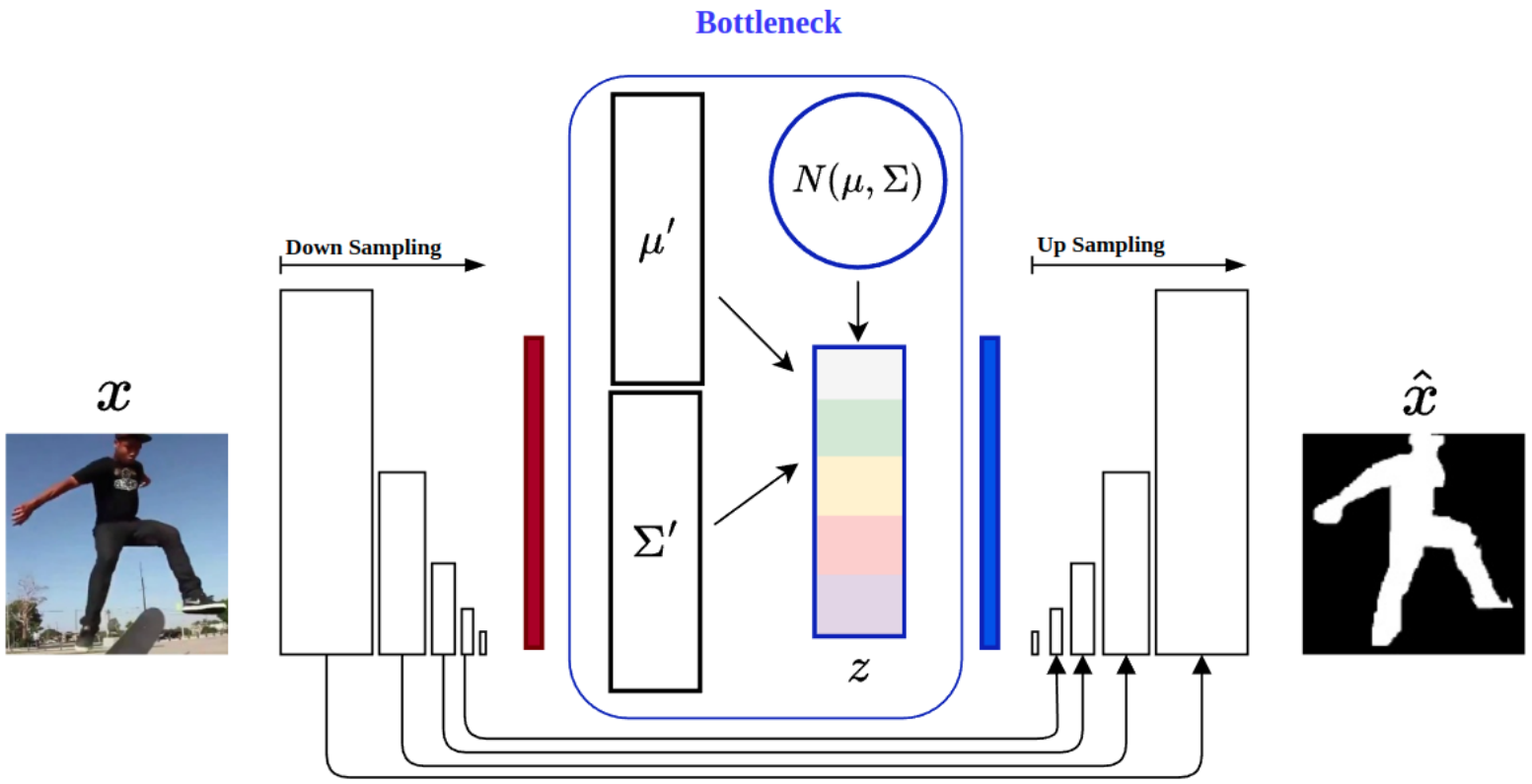
β-MuliVariational AutoEncoder: Rep. trick

lower and upper bounds of each distribution. The coefficients are calculated as $a_i = \frac{\sigma'_i}{\sigma_i}$ and $b_i = \mu'_i - \mu_i \times \frac{\sigma'_i}{\sigma_i}$ for the i^{th} part of the bottleneck, therefore:

$$z_i = a_i \varepsilon_i + b_i \tag{6}$$

where z_i is forming the i-th part of our latent set of samples z. Given a sample from the prior $\varepsilon \sim N(\mu, \Sigma)$ of size $B \times 50$, we first divide it into five parts of 10 neurons. Then, for part i, the transformation in Equation 6 is applied to force the random samples to have the encoder's posterior characteristics.

β-MuliVariational UNet



Results

Method	Data	$\mathcal{D}_{Mah}$	NLL	MSE
βMVAE	training set	0.10	4.21	0.71
	validation set	0.05	5.32	0.64
	test set	0.06	-	0.65
βMVUnet	training set	$1.05 \times e^{-6}$	1.98	0.30
	validation set	$1.22 \times e^{-6}$	2.71	0.30
	test set	$0.47 \times e^{-6}$	-	0.24

Mahalanobis distance, which computes the probability distribution distances, is used as a metric to compute the closeness of the prior and the posterior as follows:

$$\mathcal{D}_{Mah} = (\mu - \mu') \times (\frac{\Sigma + \Sigma'}{2})^{-1} \times (\mu - \mu') \tag{7}$$

Results

Table 4: Object segmentation performance on the DAVIS16 dataset [51] in terms of region similarity $\mathcal{J}$ and contour similarity $\mathcal{F}$. The **ResNet18**† is the ResNet18 architecture plus the one extra convolutional layer and OF is Optical Flow.

Methods	Backbone	OF	$\mathcal{J}\&\mathcal{F}$	$\mathcal{J}\uparrow$	$\mathcal{F}\uparrow$
SegFlow [31]	ResNet101	✓	67	67.4	66.7
SAGE [32]	-	✓	40.4	42.6	38.3
βMVAE	ResNet18	×	28.3	38.4	18.3
βMVUnet	ResNet18†	×	41.6	40.4	42.9

Results: BMVUNet

Results

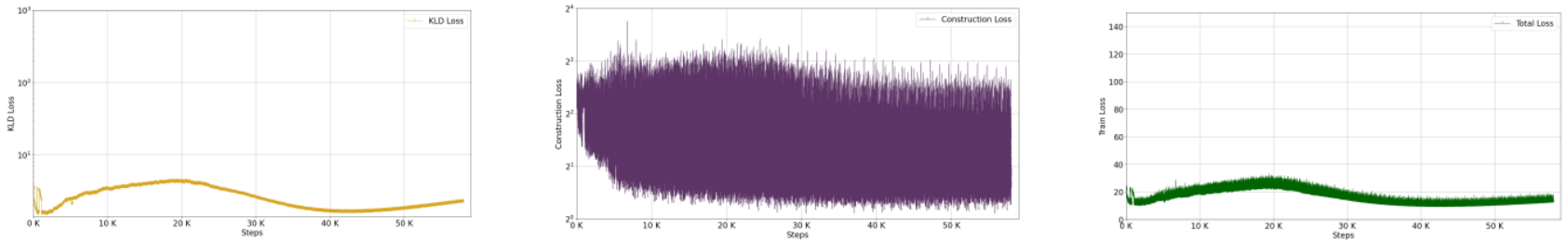


Figure 11: The loss curves in the training phase of βMVAE. Left to right: total loss, construction loss, and KL divergence loss.

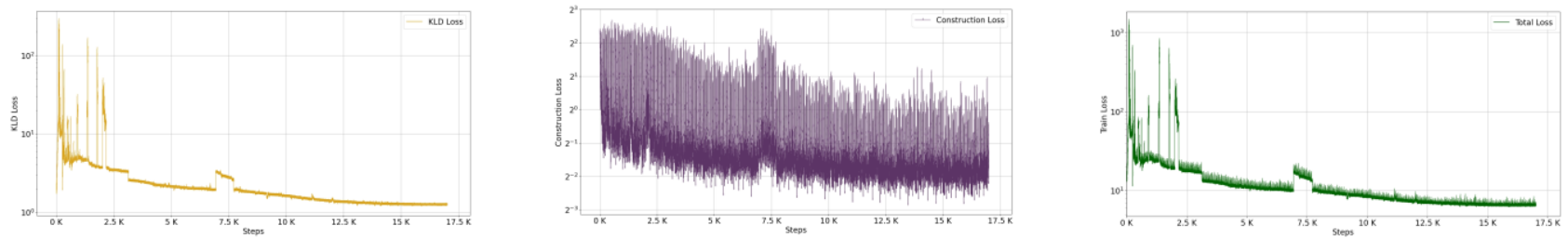
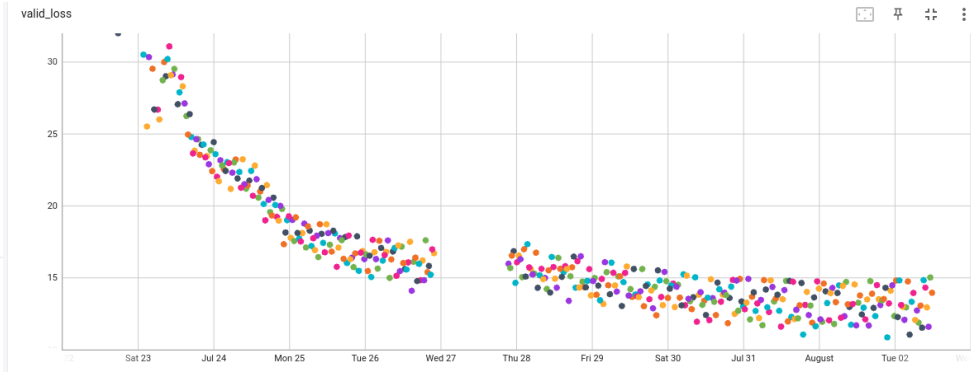


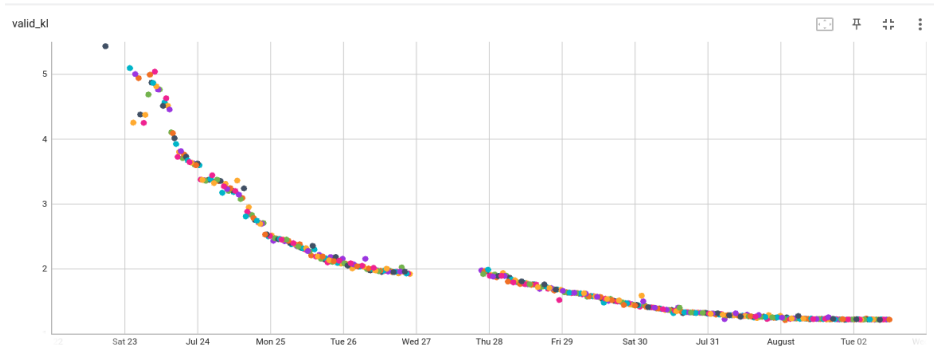
Figure 12: The loss curves in the training phase of βMVUnet. Left to right: total loss, construction loss, and KL divergence loss.

Results: Validation phase

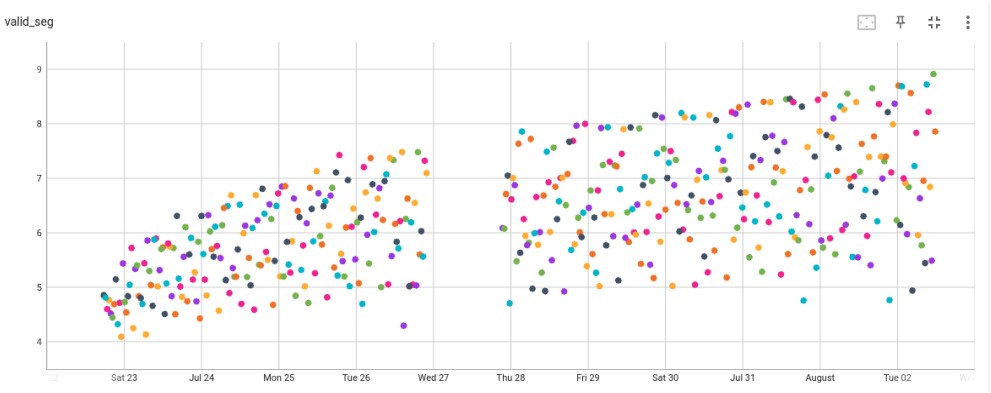
- Total loss



- KL loss



- Construction loss



Conclusion

- We proposed an asymmetric idea to model a single object's motion across video frames.
- We formulated βMVAE to learn this distribution plus the segmentation mask of the moving object.
- We demonstrated that our proposed method properly learns the KL divergence of the estimated posterior and the prior.
- The decoder of βMVAE estimates the binary mask of the object in the input frame.
- The latent variables in our setting correspond to a set of tracking parameters derived from our scenario of motion modelling in the video frames.
- The variables are dependent on each other and describe the single object's motion. Their relation is formulated as a multi-variate Gaussian distribution by a mean vector and a covariance matrix.