Supplemetary Material

Learning-based Real-time Down-sampling for Scalable Decentralized Decision-Making in Bayes-Swarm Search

Aditya Bhatt, Jhoel Witter, Prajit KrisshnaKumar, Steve Paul, Souma Chowdhury

Supplement A: Scaled Loss Function to Train Down-Sampler

The proposed loss function ($L(.)$) used to train the convolutional neural network (CNN) based down-sampler sums up Sinkhorn and Nearest Neighbor Loss, expressed in Eqn. 4 of the paper. Specific to the data range in any given application, some tuning of the weights of the loss function terms may be required, i.e., when the loss function is expressed as $L(.) = w_1 S(.) + w_2 E(.)$. Here, we provide one such example of using weighted loss terms, where each weight represents a scaling of the corresponding loss term by respective reference values roughly motivated by the range they are observed to span over training processes. Thus the loss function with scaled-weighted loss terms can be expressed as:

$$L(\mathbf{D}_M, \mathbf{D}_N) = S(\mathbf{D}_M, \mathbf{D}_N) + E(\mathbf{D}_M, \mathbf{D}'_M)$$

The resulting training history of the new CNN (called *CNN-weighted*)is shown in Fig. S1. Figure S2 shows the testing performance of the Bayes-Swarm with CNN-weighted vs. Bayes-Swarm with CNN used in the paper (that did not have loss term scaling or weighting) . Overall, their performances are comparable without any clear winner when assessed across various Environments and swarm size scenarios, as seen from Fig. S2.

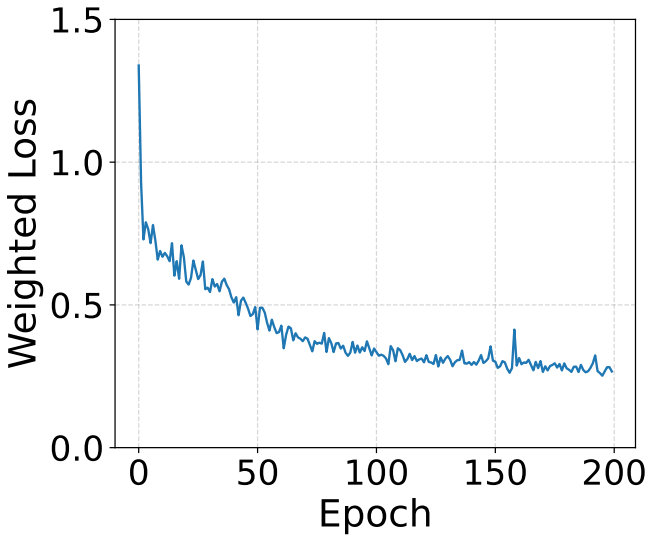
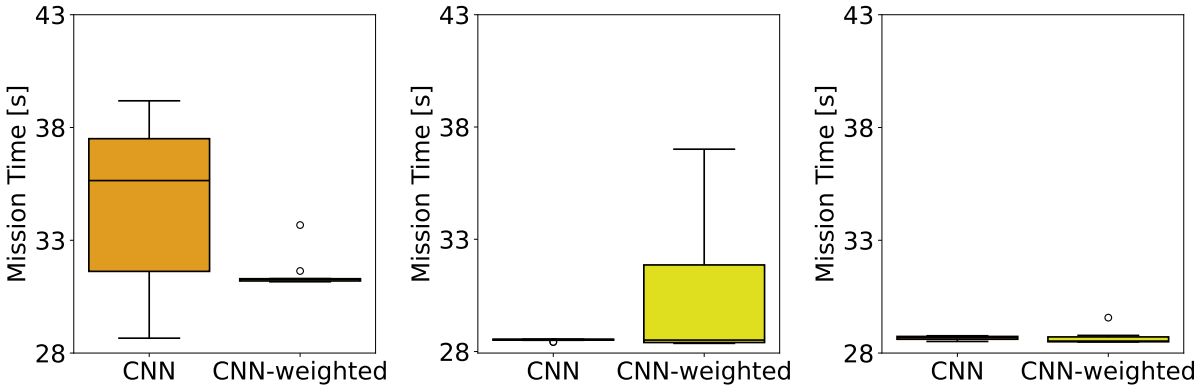
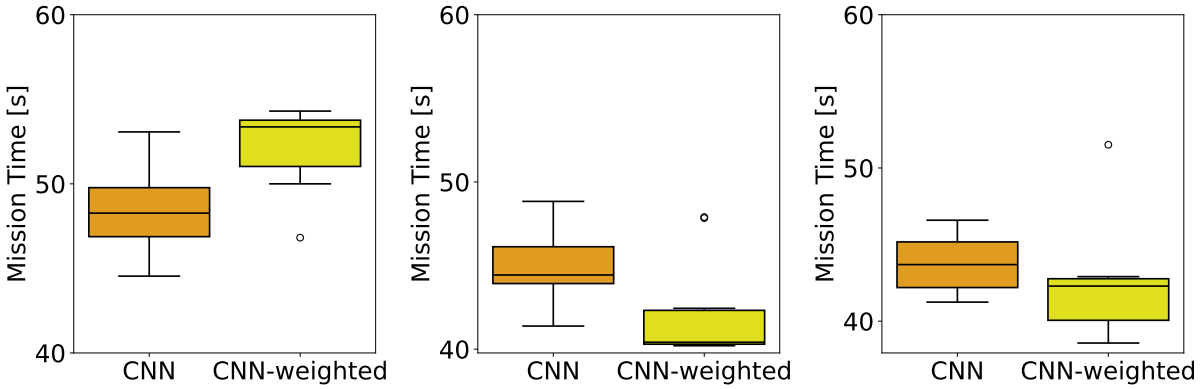


Figure S1: Weighted Loss training history



(a) 20 robots, Env 1 (b) 35 robots, Env 1 (c) 50 robots, Env 1

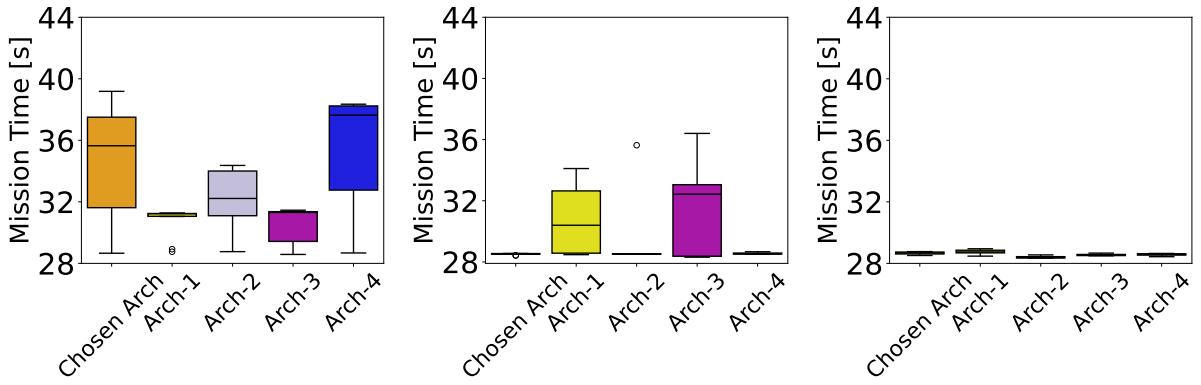


(d) 20 robots, Env 2 (e) 35 robots, Env 2 (f) 50 robots, Env 2

Fig. S2: Test results with the proposed versus the scaled loss function

Supplement B: CNN Architecture Comparisons

Here we study the sensitivity of the down-sampling performance to variations in CNN architectures in terms of the resulting CNN-Bayes-Swarm performance. To this end, we train four additional CNN architectures with higher and lower numbers of convolutional and fully connected layers compared to that in the chosen CNN used in the paper. The details of these architectures are given in Tables 1a, 1b, 2a and 2b. We then test Bayes-Swarm performance with these separate CNNs as the down-sampler. Their performances (Arch-1, Arch-2, etc.) over 10 test runs each on Environments 1, 2 and 3 are shown below, in comparison to the chosen architecture used in the paper (Chosen-Arch). Overall, the results are mixed. Except for the 20 robots case, the chosen architecture performs at par or better than the other architectures.



(a) 20 robots (b) 35 robots (c) 50 robots

Fig. S2: Performance of alternate architectures (Arch-i's) on Env 1, SSR = 1Hz, robot speed 1 m/s, compared to the chosen architecture used in the paper (Chosen-Arch).

(a) Arch-1

Layer	In Size	Out Size	Kernel
Conv2d	3x100x100	16x100x100	11x11
Conv2d	16x100x100	16x100x100	9x9
MaxPool2d	16x100x100	16x50x50	2x2
Conv2d	16x50x50	32x50x50	3x3
Conv2d	32x50x50	32x50x50	3x3
MaxPool2d	16x50x50	16x25x25	2x2
Conv2d	32x25x25	16x25x25	3x3
Conv2d	16x25x25	8x25x25	3x3
FC	8x25x25	5000	
FC	1024	200	

(b) Arch-2

Layer	In Size	Out Size	Kernel
Conv2d	3x100x100	8x100x100	11x11
MaxPool2d	8x100x100	8x50x50	2x2
Conv2d	8x50x50	8x50x50	9x9
MaxPool2d	8x50x50	8x25x25	2x2
Conv2d	8x25x25	16x25x25	3x3
FC	8x25x25	5000	
FC	5000	1024	
FC	1024	1024	
FC	1024	200	

(a) Arch-3

Layer	In Size	Out Size	Kernel
Conv2d	3x100x100	8x100x100	11x11
Conv2d	8x100x100	8x100x100	9x9
MaxPool2d	8x100x100	8x50x50	2x2
Conv2d	8x50x50	16x50x50	3x3
Conv2d	16x50x50	16x50x50	3x3
MaxPool2d	16x50x50	16x25x25	2x2
Conv2d	16x25x25	16x25x25	3x3
Conv2d	16x25x25	8x25x25	3x3
FC	8x25x25	5000	
FC	5000	1024	
FC	1024	200	

(b) Arch-4

Layer	In Size	Out Size	Kernel
Conv2d	3x100x100	8x100x100	11x11
MaxPool2d	8x100x100	8x50x50	2x2
Conv2d	8x50x50	8x50x50	9x9
MaxPool2d	8x50x50	8x25x25	2x2
Conv2d	8x25x25	16x25x25	3x3
FC	8x25x25	5000	
FC	5000	1024	
FC	1024	200	