In Proceedings of the European Conference on Computer Vision (ECCV), 2018

Sidekick Policy Learning for Active Visual Exploration Supplementary Material

Santhosh K. Ramakrishnan[1] and Kristen Grauman[2]

[1] The University of Texas at Austin, Austin, TX 78712
[2] Facebook AI Research, 300 W. Sixth St. Austin, TX 78701
srama@cs.utexas.edu, grauman@fb.com*
Project page: http://vision.cs.utexas.edu/projects/sidekicks/

In this document, we provide the following additional information:

1. Architectures and implementation details.
2. Additional policy learning details.
3. Validation plots.
4. ModelNet Hard dataset construction.
5. Additional policy visualization examples.
6. Additional training time for sidekicks.

1 Architectures and implementation details

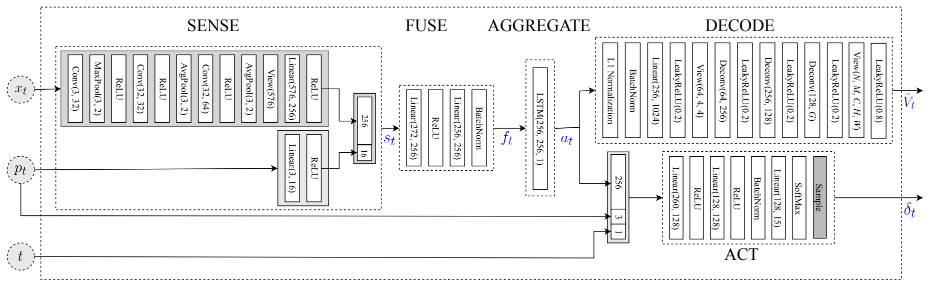

Fig. 1: Architecture for `ltla` baseline [1]. Note: G = M*N*C

Before we review the architecture, we list out some key notations:

- p_t - proprioception input, consists of the relative change in elevation, azimuth from $t-1$ to t and the absolute elevation at t.
- p^*_t - p_t augmented with absolute azimuth
- x_t - input view, dimensionality is $C \times H \times W$ where C is the number of channels, H is the image height and W is the image width. For SUN360, $C=3, H=32, W=32$ and for ModelNet Hard, $C=1, H=32, W=32$.

* *On leave from University of Texas at Austin (grauman@cs.utexas.edu)*

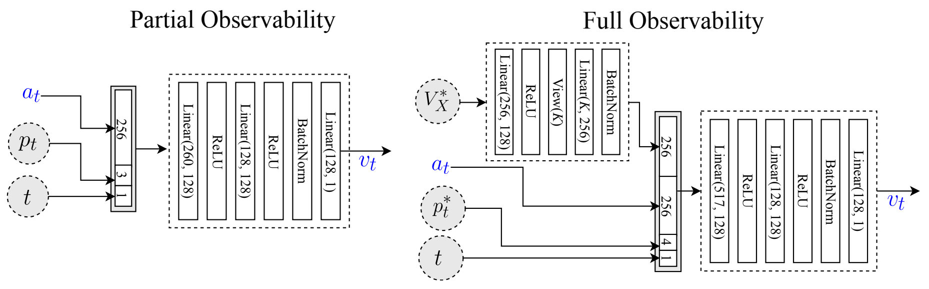

Fig. 2: Architecture for critics used in our Actor Critic training. The Partial Observability critic is used for `ours(rew)+ac`, `ours(demo)+ac` and the Full Observability critic is used for `asymm-ac` [2]. Note: $K = N * M * 128$

- M - number of azimuths in X (8 for SUN360 and 9 for ModelNet Hard).
- N - number of elevations in X (4 for SUN360 and 5 for ModelNet Hard).

We follow the same architecture (see Fig. 1) for the modules described in [1]. Models are implemented in PyTorch and layer naming conventions are accordingly followed [3]. For all the Conv layers, filter size = 5, stride = 1 and zero padding = 2; for all the Deconv (aka transposed convolution) layers, filter size = 5, stride = 2, zero padding = 2 and output padding = 1.

We have two critic architectures for our experiments (see Fig. 2). The critic with partial observability consists of a similar architecture as the ACT module. The critic with full observability takes in the absolute position on the viewgrid and the entire viewgrid as additional inputs. Each view of the viewgrid is processed by the SENSE module (to give V_X^*) and the encoded views are fused together using two FC layers. This aggregated state, proprioception input, absolute position, and fused viewgrid are concatenated and processed by the critic to obtain the value of the current view.

We use the Adam optimizer with a learning rate of $0.0001 - 0.003$, weight decay of 1e-6, and other default settings from PyTorch [4]. We also set $\lambda_r = 1$ and $\lambda_p = 1$ based on grid search. In the case of the demonstration-based sidekick, we decay T_{sup} from $T-1$ to 0 after every 50 epochs. For the reward-based sidekick, we decay the rewards by a factor of $2-10$ after every $100-500$ epochs (selected based on grid search). All the models are trained for 1000 epochs. For the reward-based sidekick, we use a non-maximal suppression neighborhood of 1 and $K = 4$ views for SUN360, and neighborhood of 2 and $K = 5$ views for ModelNet Hard. The neighborhood and number of views were selected manually upon brief visual inspection to ensure sufficient spread of rewards on the viewgrid.

To solve for Δa^* from Eq. 10 in the main paper, we use stochastic gradient descent with learning rate of 0.0001, weight decay of 0.1 and momentum of 0.9. We run the optimization for a maximum of 200 iterations, and perform early

[3] refer to http://pytorch.org/docs/master/nn.html
[4] refer http://pytorch.org/docs/master/optim.html

stopping if $|\Delta a|$ crosses $0.75|a_t|$. The parameters were selected to increase the chances of the probability change being maximised.

2 Additional policy learning details

Let the weights of the critic be denoted by W_c. Following standard actor-critic training, a regression loss over the critic's value prediction is additionally used to update the agent's parameters, specifically, W_s, W_f, W_r, W_c:

$$\Delta W_{\{s,f,r,c\}} = -\nabla_{\{s,f,r,c\}} \frac{1}{n} \sum_{i=1}^{n} \sum_{t=1}^{T-1} \left(v_t^i - \sum_{t'=t}^{T-1} r_{t'}^i \right)^2, \tag{1}$$

where n is the number of data samples and v_t^i is the value estimated by the Value network at time t for the i^{th} data sample. We additionally include a standard entropy term to promote diversity in action selection and avoid converging too quickly to a suboptimal policy. The loss term and the corresponding weight update (on W_a, W_r, W_f, W_s) are as follows:

$$\begin{aligned} L_{ent} &= \frac{1}{n} \sum_{i=1}^{n} \sum_{t=1}^{T-1} \left(\sum_{\delta \epsilon \mathcal{A}} \pi(a_t^i, j) \log\, \pi(a_t^i, \delta) \right) \\ \Delta W_{\{a,r,f,s\}} &= -\nabla_{\{a,r,f,s\}} L_{ent}. \end{aligned} \tag{2}$$

3 Validation plots

Fig. 4 in the main paper shows the validation error plots for both datasets to compare the speed of learning for our method trained with REINFORCE vs. `ltla` [1] trained with REINFORCE. Here, Fig. 3 shows the parallel validation error plots comparing `ours(rew)`, `ours(demo)` and `asymm-ac` [2] using Actor-Critic. Note that separating by REINFORCE vs. Actor-Critic ensures both sets of plots are apples-to-apples. The bump in the yellow curve on the SUN360 plot reflects how the demonstration schedule changes over epochs.

4 ModelNet Hard construction

As noted in the paper, we altered the sampling angles, lighting conditions, and object materials to increase the reconstruction difficulty of the rendered images. In Fig. 4, we render the same object using settings similar to [1] and our settings from ModelNet Hard.

The rendering details are as follows. We sampled the angles at intervals of 40° (as opposed to 30° in [1]) to reduce the number of views which were similar in appearance and geometry. We further altered the lighting positions to be non-uniform across views and used higher specularity to generate complex renderings. Specifically, we use two light sources, each placed below and above the object.

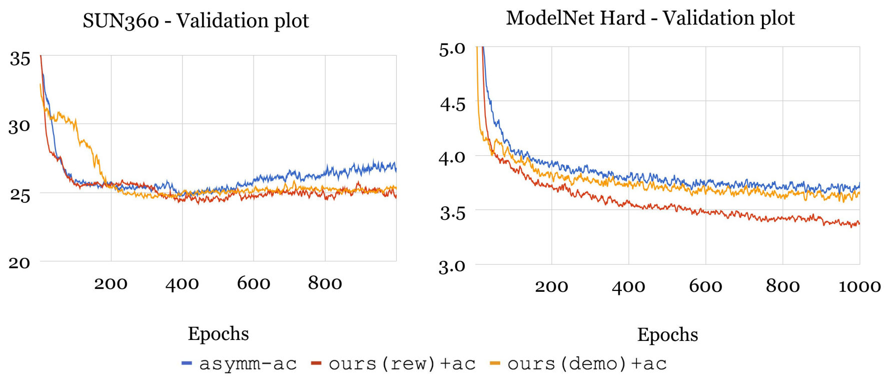

Fig. 3: Validation errors (×1000) vs. epochs on SUN360 and ModelNet Hard (Actor Critic methods)

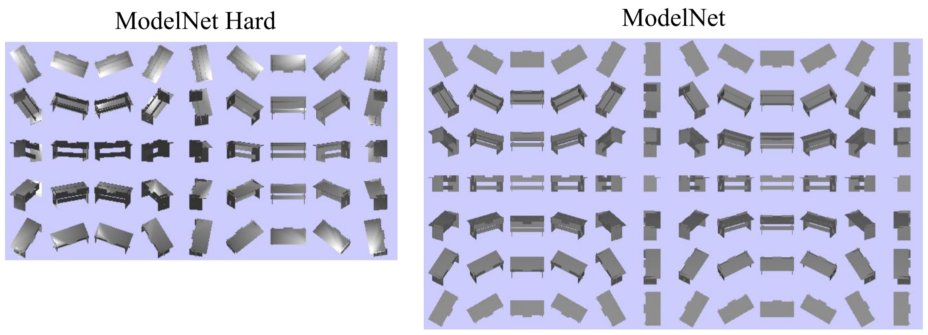

Fig. 4: An example to qualitatively compare the renderings of ModelNet Hard vs. ModelNet from [1]

The exact coordinates are selected relative to the size of the object. Each light source is placed randomly at one out of two locations for a given object. Using MATLAB's rendering toolbox[5], we render the objects with "interp" shading, "dull" material, "gouraud" face lighting, ambient strength of 0.4, diffuse strength of 0.9, specular strength of 0.7 and specular exponent of 15. The data is available to ensure reproducibility [6].

5 Policy visualization examples

Fig. 5 visualizes the policy beliefs for `ours(demo)`, `ours(rew)` and `ltla` on the SUN360 examples from the main paper and an additional example. Fig. 7 shows examples for ModelNet Hard.

[5] refer to https://www.mathworks.com/help/matlab/visualize/lighting-overview.html

[6] `http://vision.cs.utexas.edu/projects/sidekicks/`

Fig. 5 shows how all the models follow a similar behaviour of visiting regions with low heatmap densities, as indicated by the red arrows. This shows how the agents are often moving towards the views that are not yet contributing effectively to their action selection, to improve their understanding of the scene. The heatmap "density" serves as a high-level visual for the spread of the agent's reasoning about its belief state as it influences its action selection: the greater the spread, the more its belief about the full unobserved environment is directing camera motion selections. The heatmap density of `ours(demo)` lies between that of `ours(rew)` and `ltla`, which is consistent with the quantitative performance observed (refer to the main paper). We also note the qualitative difference between the heatmaps of `ours(demo)` and `ours(rew)`. While both have dense heatmaps across the entire viewgrid, `ours(demo)` appears to rely significantly more on its beliefs about the ground plane of the scene. However, there are cases where the visualizations are not conclusive in differentiating between the policies. As shown in Fig. 6, we can see that visualizations are dense across all models, and therefore, less conclusive.

In Fig. 7, we see our visualization is less effective in differentiating the policies on ModelNet Hard, possibly due to the narrower margins in the reconstruction errors for this dataset. However, it is interesting to note that the heatmap densities are better concentrated on the object for `ours(rew)` and `ours(demo)`, whereas it often unnecessarily leaks to the background pixels for `ltla`.

6 Additional training time for sidekicks

In order to account for additional training time required to train the sidekicks, we analyze the time taken for training various models and sidekicks. Since *all* models are pretrained with $T = 1$, including `ltla` — the training overhead ($\sim$ 64 min) is identical for the baseline. Both sidekicks use the $T = 1$ model to compute scores (see Sec. 3.4 from main paper), a one-time cost of $\sim$ 10 min. To train for 500 epochs, `ours (rew)` and `ours (demo)` require $\sim$ 450 and $\sim$ 485 min, resp, while `ltla` and `asymm-ac` take $\sim$ 465 and $\sim$ 529 min, resp (averaged over 3 runs) [7]. Therefore, the additional training time for sidekicks is nominal in comparison to the overall training process. However, training the expert for `expert-clone` takes as long as it takes to train a full model ($\sim$ 1000 minutes for 1000 epochs), which is $\sim 17\times$ the time required to pre-train at $T = 1$ and pre-compute the sidekick scores.

References

1. Jayaraman, D., Grauman, K.: Learning to look around: Intelligently exploring unseen environments for unknown tasks. In: Computer Vision and Pattern Recognition, 2018 IEEE Conference on. (2018)
2. Pinto, L., Andrychowicz, M., Welinder, P., Zaremba, W., Abbeel, P.: Asymmetric actor critic for image-based robot learning. Robotics: Science and Systems (2018)

[7] Experiments were run on a Intel(R) Xeon(R) CPU @ 1.70GHz system with GeForce GTX 1080 GPU.

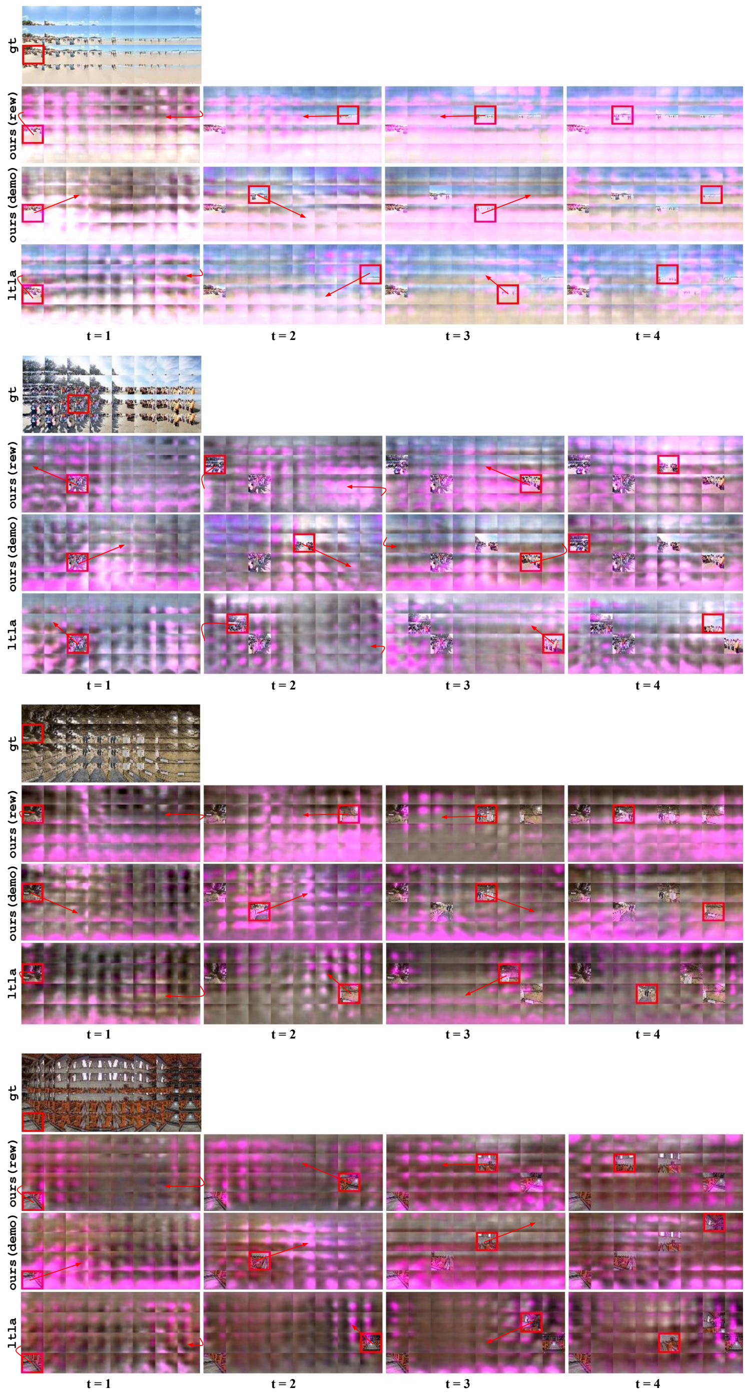

Fig. 5: Policy visualizations of `ltla`, `ours(rew)` and `ours(demo)` on four examples from SUN360. The policies tend to visit regions on the viewgrid with low heatmap densities in order to improve their belief about the environment. Better policies tend to more rapidly improve their beliefs, as witnessed by denser heatmaps. Best viewed on pdf with zoom.

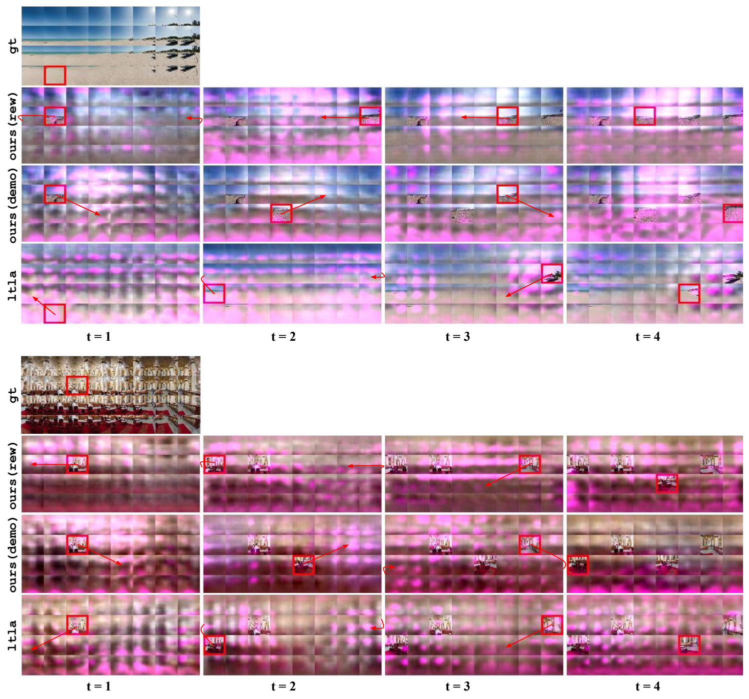

Fig. 6: Examples of less conclusive visualizations on SUN360, where `ltla`, `ours(rew)` and `ours(demo)` have similar heatmap densities. Best viewed on pdf with zoom.

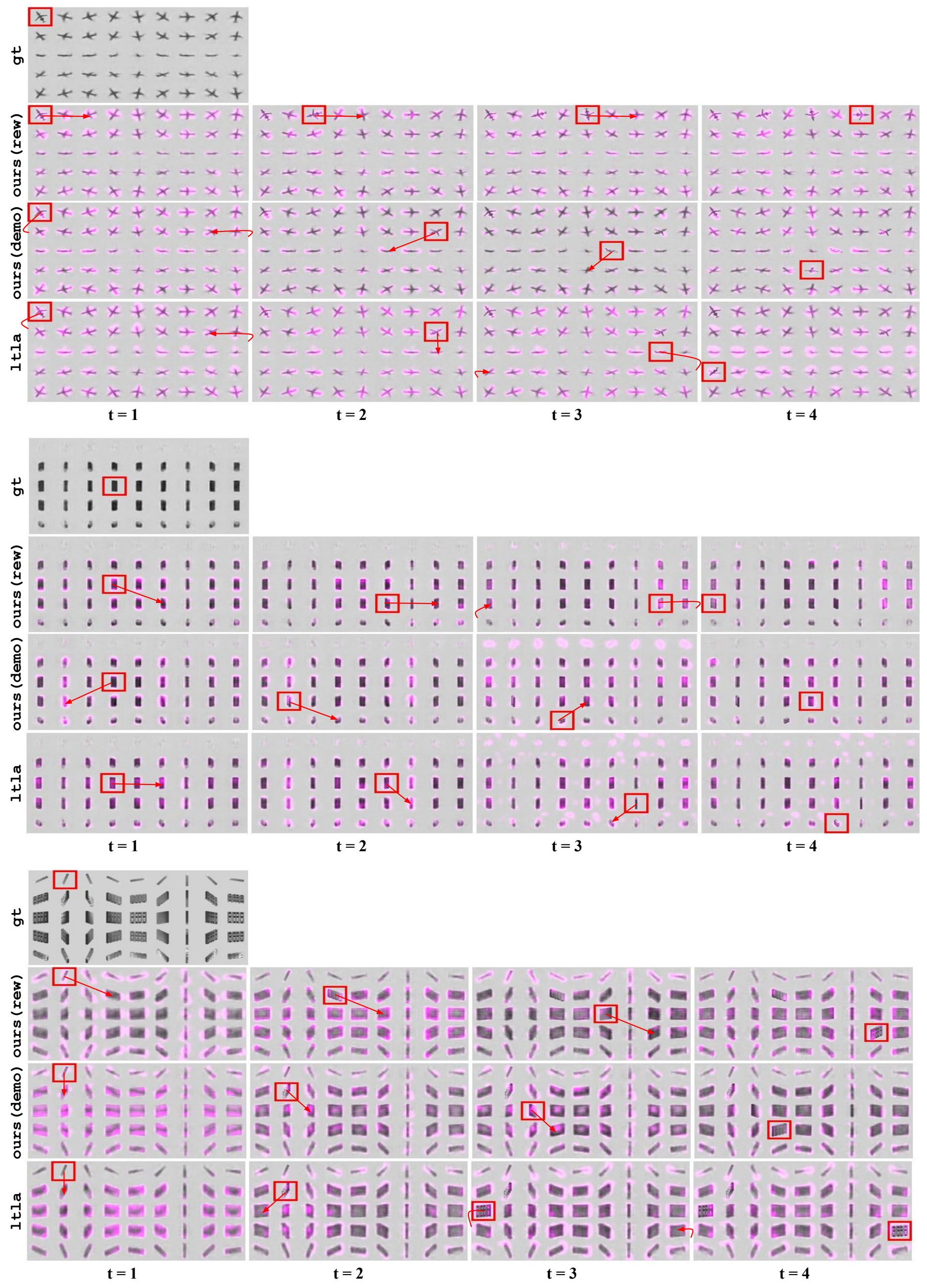

Fig. 7: Policy visualizations of `ltla` and `ours(rew)` on three examples from ModelNet Hard. Best viewed on pdf with zoom.