Appendix

A. Different Detector Architecture.

To validate that our approach is not intrinsically dependent on the DETR [5] object detector architecture, we train a version of INTERACTRON Faster-RCNN. The baseline Faster-RCNN detector achieves an AP_{50} of 0.44, while using INTERACTRON achieves an AP_{50} of 0.58. While we are able to obtain better performance using the DETR object detector backbone, we show that INTERACTRON can work with other detector architectures as well.

B. Pseudo Labels.

We investigate the viability of using pseudo-labels instead of a learned loss using the Multi-frame model and a very simple pseudo-label method. We take the pre-trained Multi-frame model, and during evaluation apply a gradient update using the detections as labels, weighed by the detection confidence. This gives us a small performance improvement of less than 1 point with an AP_{50} of 0.549. While our learned loss still outperforms this model, this ablation shows that a pseudo-label approach could also be viable in our problem setting.

C. Model Details

We use the DETR [5] model with a ResNet50 backbone for our detector. We freeze the backbone and Transformer encoder and only meta-train the decoder portion of DETR. We use the loss function based on the Hungarian matching algorithm described in the DETR paper as our ground truth detection loss $\mathcal{L}_{det}$. We set a maximum of 50 detections per image frame.

For our supervisor we use a 4 layer, 8 head Transformer based on the GPT architecture with an internal dimension of 512. Our supervisor also consists of two separate embedding layers for the Image Features and Object Detection Features respectively. What we refer to as "Object Detection Features" are described as the output embeddings of the query tokens in the DETR paper. A positional embedding is learned for each token in the sequence. The supervisor also contains two decoders that consists of three consecutive linear layers each, with a hidden dimension of 512, separated by the GeLU non-linearity. One of the decoders consumes the outputs of the detection tokens and is used to compute the learned loss, while the other is used to decode the output of the Policy Tokens and produce a policy.

The learned loss is computed by taking the ℓ_2 norm of the outputs of all the Detection Tokens passed through the decoder.

Figure 4 contains a diagram of the INTERACTRON pipeline with tensor dimensions.

D. Data Collection Details

D.1. iTHOR Data Collection Details

For all of the iTHOR datasets the objects are labeled as one of the 1,235 classes which comprise the union of all the LVIS object categories and all the iTHOR object categories.

The iTHOR Pre-Training Dataset is collected from the 100 scenes in the train and val splits (scenes 0-25, 200-225, 300-325, 400-425). We uniformly sample frames from these scenes to collect a total of 10,000 frames and their annotations. A random shuffle of the placement of objects in the scene is performed before each sample is collected. If a frame has less than 3 objects visible in it, it is rejected and a new sample is drawn. The annotations are stored in LVIS format. The data is added to the LVIS dataset to collectively form the iTHOR+LVIS pre-training dataset.

The iTHOR Training Set is collected from the 100 scenes in the train and val splits. We uniformly sample starting positions from these scenes to collect a total of 1,000 starting location frames. A random shuffle of the placement of objects in the scene is performed before each sample is collected. If a frame has less than 3 objects visible in it, it is rejected and a new sample is drawn. In addition to this, every frame in every possible trajectory the agent could take from any sampled starting location is also collected. This implementation detail allows us to pre-cache every possible trajectory the agent could take to improve training efficiency.

The iTHOR Test Set is collected from the 20 scenes in the test split (scenes 25-30, 225-230, 325-330, 425-430). We uniformly sample starting positions from these scenes to collect a total of 100 starting location frames. If a frame has less than 3 objects visible in it, it is rejected and a new sample is drawn. In addition to this, every frame in every possible trajectory the agent could take from any sampled starting location is also collected. This implementation detail allows us to pre-cache every possible trajectory the agent could take to improve training efficiency.

D.2. Habitat Data Collection Details

For all of the Habitat datasets the objects are labeled as one of the 1,255 classes which comprise the union of all the LVIS object categories, all the iTHOR object categories and all the Habitat object categories.

The Habitat Pre-Training Dataset is collected from the 56 scenes in the MP3D train split. We uniformly sample frames from these scenes to collect a total of 10,000 frames and their annotations. Each scene has an equal number of samples, regardless of its size. If a frame has less than 3 objects visible in it, it is rejected and a new sample is drawn. The annotations are stored in LVIS format. The data is added to the iTHOR+LVIS pre-training dataset to collectively form the Habitat+iTHOR+LVIS dataset.

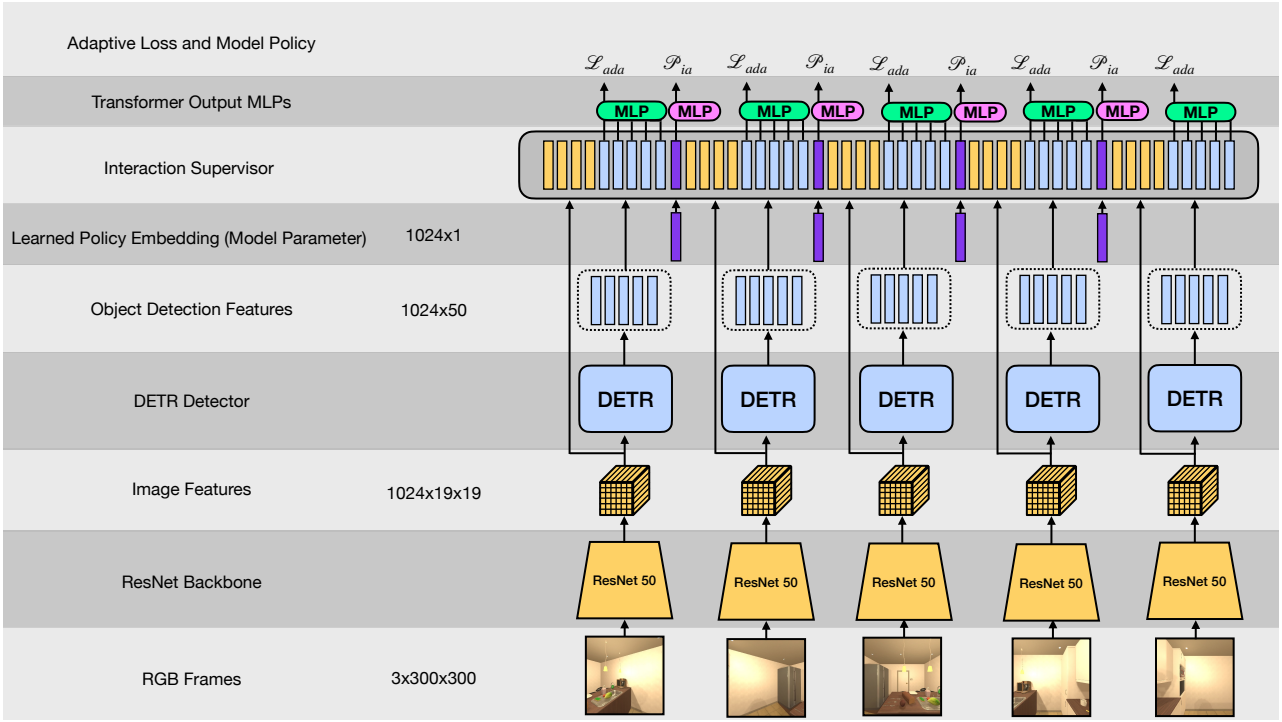

Figure 4. The detailed architecture of the INTERACTRON model. The first column describes each layer, while the second column provides the dimensions of the intermediate tensors where applicable.

The Habitat Test Dataset is collected from the 11 scenes in the MP3D val split. We uniformly sample starting positions from these scenes to collect a total of 100 starting location frames. Each scene has an equal number of samples, regardless of its size. If a frame has less than 3 objects visible in it, it is rejected and a new sample is drawn. In addition to this, every frame in every possible trajectory the agent could take from any sampled starting location is also collected. This implementation detail allows us to precache every possible trajectory the agent could take to improve training efficiency.

E. DETR Pre-Training Details

We pre-train the DETR object detector using the official DETR codebase, modified to accept the format of our dataset. We use the same training parameters as proposed in the original DETR paper to train one detector on the iTHOR+LVIS pre-training dataset and another on the Habitat+iTHOR+LVIS dataset. We scale all of the pre-training images such that one side has a dimension of 300.

F. Training Details

F.1. Interactron Training

The INTERACTRON model is trained for 1,000 epochs with the AdamW optimizer, using an initial learning rate of $3e{-}4$ and a linearly annealing schedule. Both the supervisor model producing the learned loss and policy and the detector itself are trained using these parameters. The training takes approximately 23 hours using a single Nvidia RTX 3090.

During training we do not follow the policy of the model, but rather explore every possible trajectory the agent could take, from a given starting location. We then record the $IFGA$ value for each possible trajectory and update it whenever we revisit the same trajectory. We optimize our policy to always pursue the trajectory with the lowest recorded $IFGA$.

In slight contradiction to common nomenclature, one epoch does not represent a pass through every single frame in our dataset, as the agent can explore multiple trajectories form each starting location. For our 5 frame (4 step) experiments, for example, we can only guarantee that every possible frame in the dataset has been explored after 256 epochs of training.

F.2. Interactron 7 Frame Training

For training the 7 frame model (where the agent takes 6 steps) we utilize a training regime of 3,000 epochs using the same initial learning rate and annealing schedule. This way we ensure that every possible trajectory the agent could take from each initial frame is explored multiple times.

F.3. Interactron 9 Frame Training

For training the 9 frame model (where the agent takes 8 steps) we utilize a training regime of 9,000 epochs using the same initial learning rate and annealing schedule. This way we ensure that every possible trajectory the agent could take from each initial frame is explored at least once.