Learning to Navigate Using Mid-Level Visual Priors

Supplementary Material

The following items are provided in the supplementary material:

3rd Conference on Robot Learning (CoRL 2019), Osaka, Japan.

A Policies on Real Robots

We include a video for our example of policies generalizing to a real robot on our website.. The video shows our results from training a *curvature*-based visual-target navigation policy on a simulated Turlebot2. We then then tested the trained policy (not fine-tuned) on a real Turtlebot2. The video shows that the robot is able to successfully navigate to the correct cube, even in the presence of several distracting boxes which were not present during training[1].

Figure 1: **Turtlebot2 navigating to the hallway through a cluttered environment.** Agents trained using mid-level vision in simulation successfully navigate in cluttered real-world environments without fine-tuning. The agent's only input is the RGB frame shown in the top right. See the included video for more.

B Sample Complexity

As described in the main paper, we measure sample complexity with respect to the number of training frames and the number of sample clusters (buildings).

Performance by Number of Training Frames Our methods are able to reach a similar performance as other methods in much fewer samples, as seen in Fig. 2.

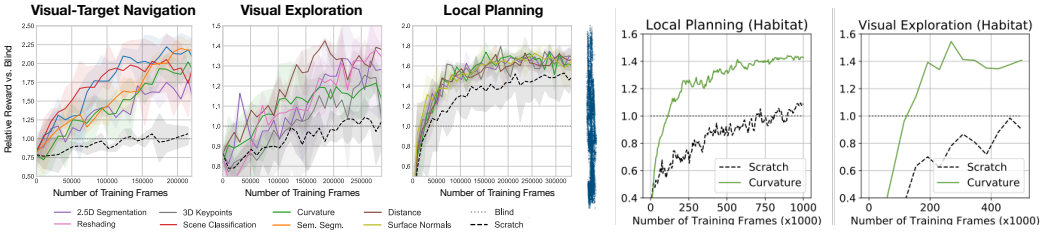

Figure 2: **Sample Complexity (Number of Frames).** Plots show average rewards relative to blind in the test environment with varying amounts of training data for both features and scratch. Feature-based policies learn faster and achieve higher final performance.

Performance by Number of Sample Clusters (buildings) In Fig. 3, we show that feature-based agents can achieve higher reward even when training with fewer buildings compared to agents

[1]We thank Gene Lewis for providing the video.

trained from scratch. Performance continues to improve with more clusters (buildings). Note that scratch is unable to improve performance with additional buildings, demonstrating that it has overfit to the training buildings.

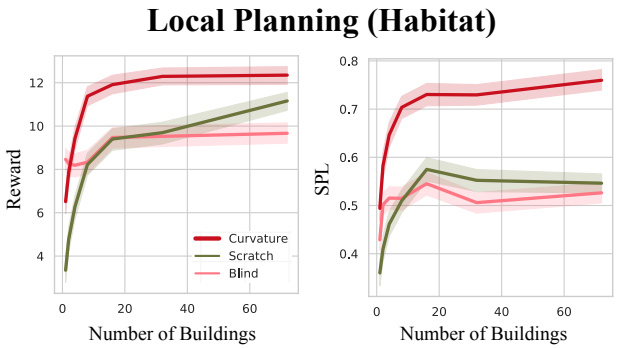

Figure 3: **Sample Complexity** Agents with mid-level vision need fewer buildings to learn useful behaviors compared to scratch and blind agents.

C Universality Experiments

C.1 Universality in Additional Buildings (Gibson)

In order to rule out that our test environment was in some way anomalous, we repeated our testing in 9 additional environments. Figure 4 shows that the rewards in our main test are extremely highly correlated with the rewards in the alternate test environments. The feature rankings are extremely strongly correlated between the main test environment and the alternate test environments with a Spearman's ρ of 0.93 (Navigation) and 0.85 (Exploration).

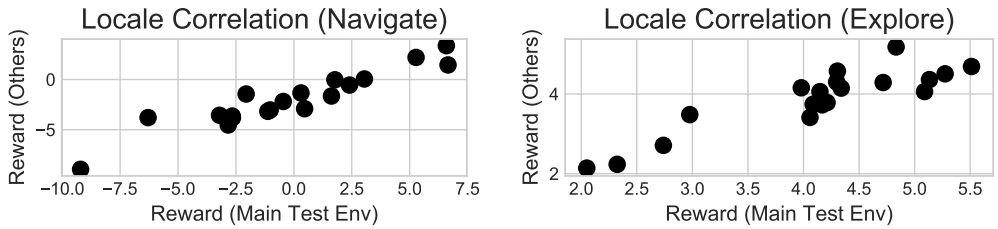

Figure 4: **Correlation between main Gibson test environment and alternates.** The reward in the main Gibson test environment (x-axis) and the reward averaged over 9 other test environments (y-axis). Each point marks the reward in both environments for a single feature (averaged over several seeds). Results are presented for both visual-target navigation (left) and visual exploration (right).

C.2 Universality in Additional Simulators

To evaluate whether our findings are an artifact of any specific environment, we tested in an additional environment by implementing navigation and exploration in other 3D simulators, VizDoom [1] and Habitat. We found that features which perform well in Gibson also tend to perform well in other simulators. We also replicated our rank reversal findings (with high confidence), including the geometric/semantic distinction for exploration/navigation and the lack of a *universal feature*. Here, too, maximizing the combined score requires choosing the third- or fourth-best feature for any given task.

D Experimental Setup

D.1 Architectures

Our architecture is defined by representation followed by policy, depicted in Figure 5.

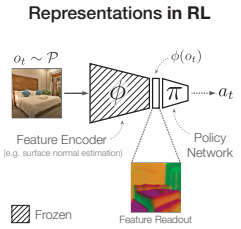

Figure 5: **Architecture.** Representation followed by Policy

D.1.1 Representation

The methods we use for representation have the following networks:

- **Pretrained Feature encoder:** For all tasks, we modified a ResNet-50 encoder with no average-pooling and replace the last stride 2 convolution with stride 1. This gives us an output shape of $16 \times 16 \times 2048$. we use a 3×3 convolution to transform the output of shape $16 \times 16 \times 8$. Note we share pretrained feature weights across all frames.
- **Feature readout network:** The feature readout network maps the feature encoding to a final representation. It consists of one convolutional (Conv) layer and two fully-connected (FC) layers:
  1. Conv, 32 channel, 4x4 kernel, stride 4
  2. FC, ouptut size = 1024
  3. FC, ouptut size = 512
- **Atari-net network:** The atari-net network consists of three convolutional layers:
  1. Conv, 32 channel, 8x8 kernel, stride 4
  2. Conv, 64 channel, 4x4 kernel, stride 2
  3. Conv, 32 channel, 3x3 kernel, stride 1
  4. FC, output size = 512

Putting it together

Our *mid-level* approach computes representation by first obtaining features via the pretrained feature encoder then passing it through the feature readout network.

The *tabula rasa* approach downsamples the image to 84×84 then feeds the input through the Atari-net network.

We also run ablation studies on the aforementioned networks.

Incorporating Additional Task-Specific Information

For *Visual Exploration*, we incorporate the occupancy grid by feeding it through a separate Atari-Net and concatenate its intermediate representation (before the FC layers) with the intermediate representation from pixels before flattening and forwarding to the FC layers.

For *Local Planning*, we incorporate the target vector by expanding each element to an (H $\times$ W) "image", and append this to the intermediate representation from pixels in the channel dimension before the last convolutional layer.

D.1.2 Policy

The representation lies in $\mathbb{R}^{512}$. We compute the value by applying a linear layer to the representation. We compute the action probabilities the same way.

D.2 Environment-Specific Task Details

Here, we provide environment specific information about all of the tasks.

D.2.1 Gibson

Local Planning: The reward for reaching the goal is $+20$. The reward for progressing to the goal is $0.1 * (d_{t-1} - d_t)$, where d_t is the distance to the goal at timestep t. The penalty for living is -0.05 and penalty for obstacle collision (-0.25). The goal is sampled from a Gaussian distribution, $\mathcal{N}(\mu = 5 \text{ meters}, \sigma^2 = 2 \text{ meters})$.

Visual Exploration: The agent receives a reward of $+0.1$ for each cell unlocked. The episode lasts for 1000 agent steps.

Visual Navigation: The agent has to navigate to a wooden crate. The agent gets a reward of $+10$ for hitting the target and a reward of -0.025 at each timestep for living.

D.2.2 Habitat

Local Planning The reward for reaching the goal is $+10$. The reward for progressing to the goal is $0.1 * (d_{t-1} - d_t)$, where d_t is the distance to the goal at timestep t. The penalty for living is -0.01 and there is no penalty for obstacle collision. The goals in both training and validation split are curated by Manolis Savva* and Batra [2] and we refer the reader there for more details.

Visual Exploration: The agent receives a reward of $+0.1$ for each cell unlocked. The episode lasts for 1000 agent steps.

D.2.3 Doom

Visual Exploration: The agent receives a reward of $+1$ for each cell unlocked. The episode lasts for 1000 agent steps.

Visual Navigation: The agent must navigate to a green torch. The agent gets a reward of $+100$ for hitting the target and a reward of -1 for living.

D.3 Train/Test Splits

The tasks are realized in the simulations Habitat, Gibson, and VizDoom. The train and test splits are shown below.

D.3.1 Habitat Train/Test Split

Figure 6: **Sample Frame from Local Planning in Habitat**. [Left] The agent observes this rgb image. [Middle] Top-down view of the map with boundary information. This map is not provided to the agent and used purely for visualization purposes [Right] From target coordinates provided at every timestep, we can reconstruct a visitation map which we provide the agent.

We use the split used in the CVPR 2019 Habitat Challenge. A sample frame is shown in Figure 6. We train on 72 buildings and test on 14 unseen buildings. The list of building names are provided on the challenge page and are also listed directly below:

Train: Rancocas, Cooperstown, Hominy, Placida, Arkansaw, Delton, Capistrano, Mesic, Roeville, Angiola, Mobridge, Nuevo, Oyens, Quantico, Colebrook, Sawpit, Hometown, Sasakwa, Stokes, Soldier, Rosser, Superior, Nemacolin, Pleasant, Eagerville, Sanctuary, Hainesburg, Avonia, Crandon, Spotswood, Roane, Dunmor, Spencerville, Goffs, Silas, Applewold, Nicut, Shelbiana, Azusa, Reyno, Dryville, Haxtun, Ballou, Adrian, Stanleyville, Monson, Stilwell, Seward, Hambleton, Micanopy, Parole, Nimmons, Pettigrew, Bolton, Sumas,

Sodaville, Mosinee, Maryhill, Woonsocket, Springhill, Annawan, Albertville, Anaheim, Roxboro, Beach, Bowlus, Convoy, Hillsdale, Kerrtown, Mif flintown, Andover, Brevort

Test: Denmark, Greigsville, Eudora, Pablo, Elmira, Mosquito, Sands, Swormville, Sisters, Scioto, Eastville, Edgemere, Cantwell, Ribera

For our generalization experiments in which we vary the number of training buildings to $N \in \{1, 2, 4, 8, 16\}$ buildings, we use the first N train buildings in the list above. When we limit training to trajectories in which the target is at most k meters away, we still use all the train buildings from above.

D.3.2 Gibson Train/Test Split

Figure 7: **Sample Frame from Local Planning in Gibson**. [Left] The agent observes this rgb image. [Middle] Top-down view of the map with boundary information. This map not provided to the agent and used purely for visualization purposes [Right] Visualization of a mid-level feature (surface normals).

In Gibson, we train in one building for both tasks (Beechwood, which is a medium-sized house) and evaluate the policies in 10 test environments given in the table below. The environments were selected for diversity. For exploration, large layouts on ground floors were selected (since the agent was trained on the ground floor). For navigation, generally open spaces were selected so the agent would need not have to take winding, complicated paths to the goal aside from very local (less than 3 m) obstacle avoidance (again, the reason for this choice being that the agent was trained in such a space). A sample frame is shown in Figure 7.

Task	Train Env.	Test Envs.
Navigation	Beechwood	Ancor Aloha Corder Duarte Eagan Globe Kemblesville Martinville Vails Area 1 (2D3DS)
Exploration	Beechwood	Aloha Duarte Eagan Globe Hanson Hatfield Kemblesville Martinville Sweatman Wiconisco

Figure 8: **Train/Test Split (Gibson).** Environments chosen for train and test split in Gibson. Area 1 is from the Stanford 2D-3D-S dataset. [3]

D.3.3 Doom Texture Split

Please see the files `texture_split/doom_train_textures.txt` for a list of train textures and `texture_split/doom_test_textures.txt` for a list of the test textures. A sample frame is shown in Figure 9.

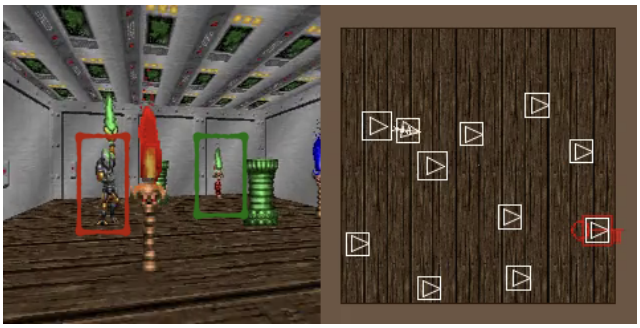

Figure 9: **Sample Frame from Visual-Target Navigation in Doom**. [Left] The agent observes this rgb image. The goal is to navigate to the green torch indicated in a green box while avoiding distractors like the one indicated in a red box. [Right] Top-down view of the map with boundary information. This map is not provided to the agent and used purely for visualization purposes.

D.4 Action Space

We assume a low-level controller for robot actuation, enabling a high-level action space of

$$\mathcal{A} = \{\texttt{turn\_left}, \texttt{turn\_right}, \texttt{move\_forward}\}.$$

The forward action is a translation in the direction of the robot's heading of 0.25 m in Habitat and 0.1 m in Gibson. The turn actions represent in-place rotations of ± 0.10 degrees in Habitat and ± 0.24 radians (about 14 degrees) in Gibson. The actions in Doom are similar to those in Gibson, and are defined by the VizDoom simulator.

D.5 Hyperparameters

A complete list of hyperparameters can be found in the folder `./hyperparameters` and `./configs`. For each task, we conducted a grid search for the best hyperparameters for "scratch". We then used these hyperparameters for all features.

For Habitat local planning, we use the following hyperparameters: learning rate of $1e-4$, entropy regularization coefficient of $1e-4$, value loss coefficient of $1e-3$, discount factor γ of 0.99, maximum gradient norm of 0.5, PPO clip parameter of 0.1, GAE τ of 0.95, replay buffer size of 3000 with each rollout being 1000 steps. We use Adam [4] optimizer and stack 4 frames per observation. Curiosity has curiosity reward coefficient of 0.1, forward loss coefficient of 0.2, and inverse loss coefficient of 0.8. SLAM uses a map size of $12m$.

For Habitat exploration, the hyperparameters are the same as Habitat local planning except for the following: learning rate of $1e-3$.

D.6 Code

We provide our code for reproducing our experiments on our website.

E Full Train/Test curves

For completion, we present the training and test curves from our experiments for all tasks, for all features and baselines, and all random seeds. See Figures 10, 11, 12, 13, 14, 15, 16, and 17.

References

[1] M. Kempka, M. Wydmuch, G. Runc, J. Toczek, and W. Jaskowski. Vizdoom: A doom-based AI research platform for visual reinforcement learning. *CoRR*, abs/1605.02097, 2016. URL http://arxiv.org/abs/1605.02097.

[2] O. Y. Z. E. W. B. J. J. S. J. L. V. K. J. M. D. P. Manolis Savva*, Abhishek Kadian* and D. Batra. Habitat: A platform for embodied ai research. *arXiv preprint arXiv:1904.01201*, 2019.

[3] I. Armeni, S. Sax, A. R. Zamir, and S. Savarese. Joint 2d-3d-semantic data for indoor scene understanding. *CoRR*, abs/1702.01105, 2017. URL http://arxiv.org/abs/1702.01105.

[4] D. P. Kingma and J. Ba. Adam: A method for stochastic optimization. *CoRR*, abs/1412.6980, 2014. URL http://arxiv.org/abs/1412.6980.

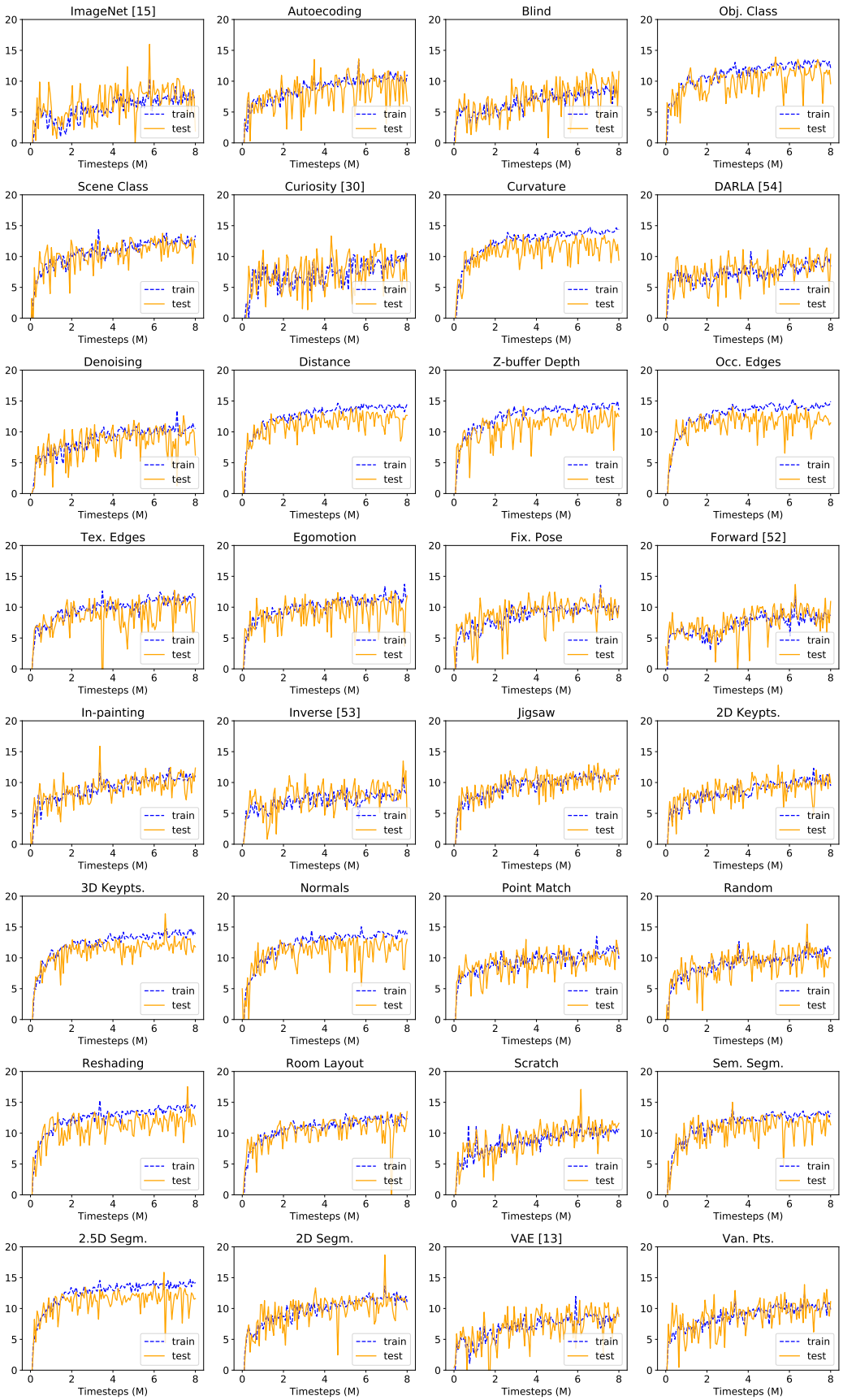

Figure 10: **Full Learning Curves (Habitat Local Planning).** Full train and test curves for all features and baselines in the navigation task in the Habitat environment.

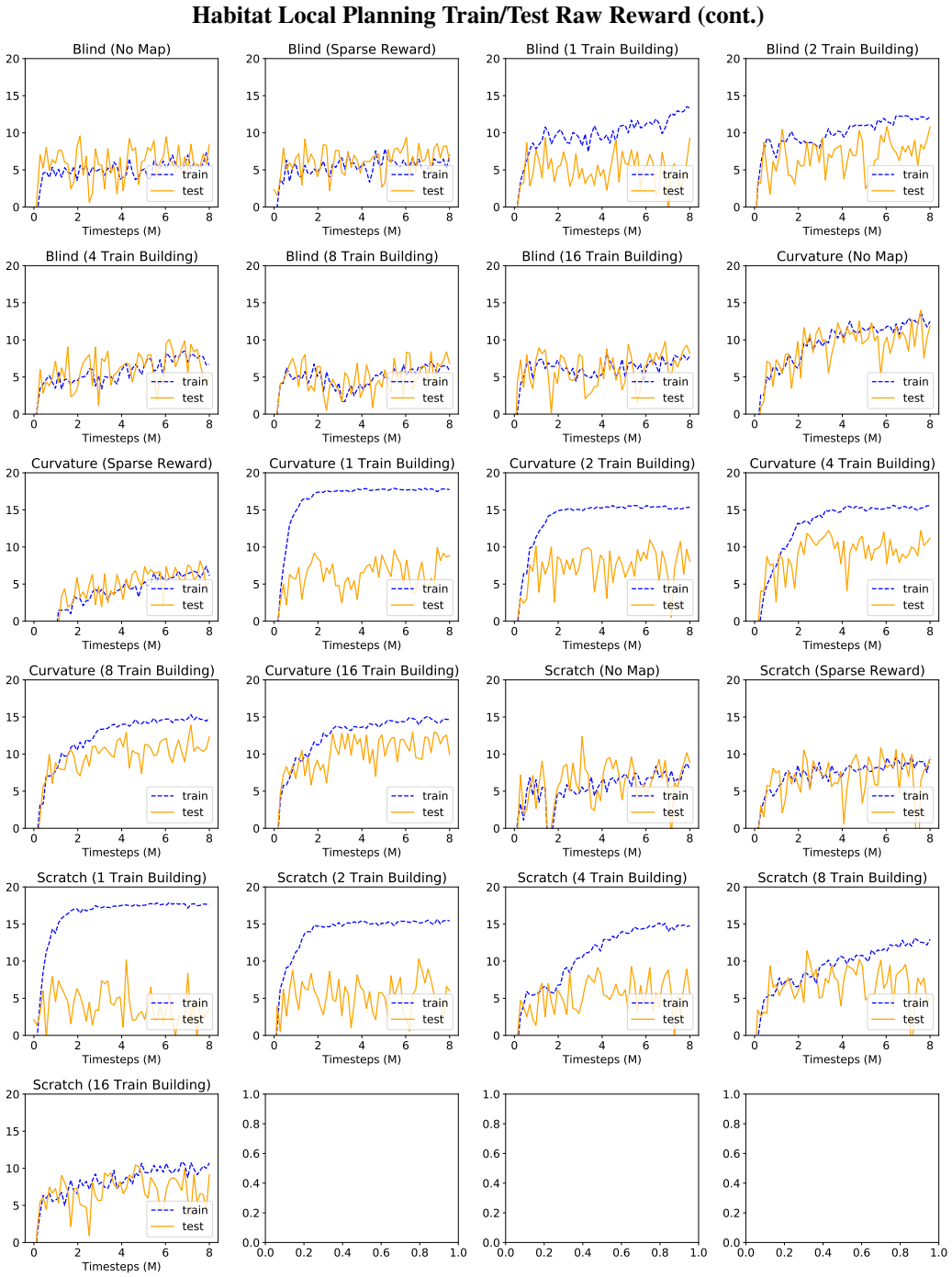

Figure 11: **Full Learning Curves (Habitat Local Planning Cont.).** Full train and test curves for all features and baselines in the navigation task in the Habitat environment.

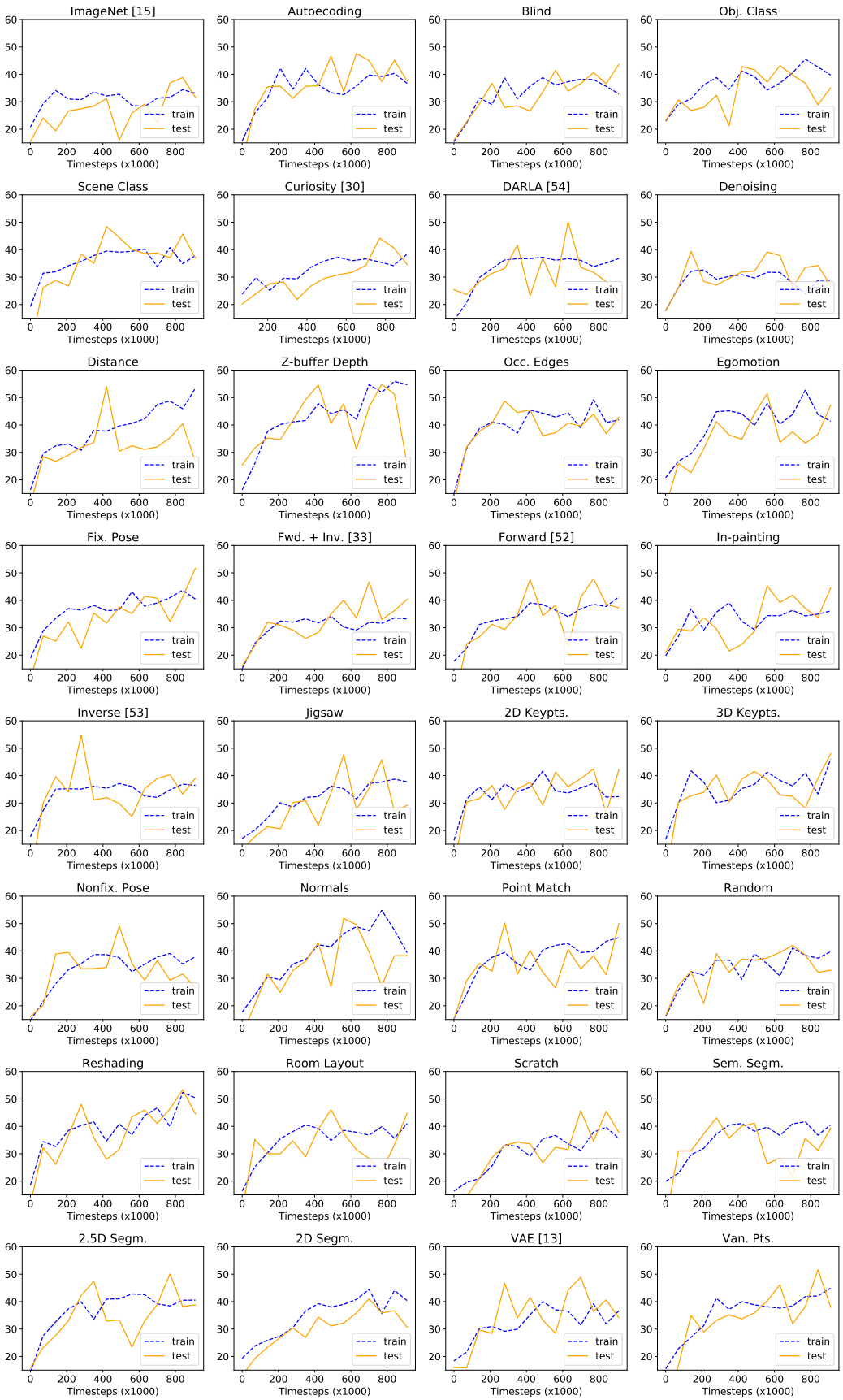

Figure 12: **Full Learning Curves (Habitat Exploration).** Full train and test curves for all features and baselines in the navigation task in the Habitat environment.

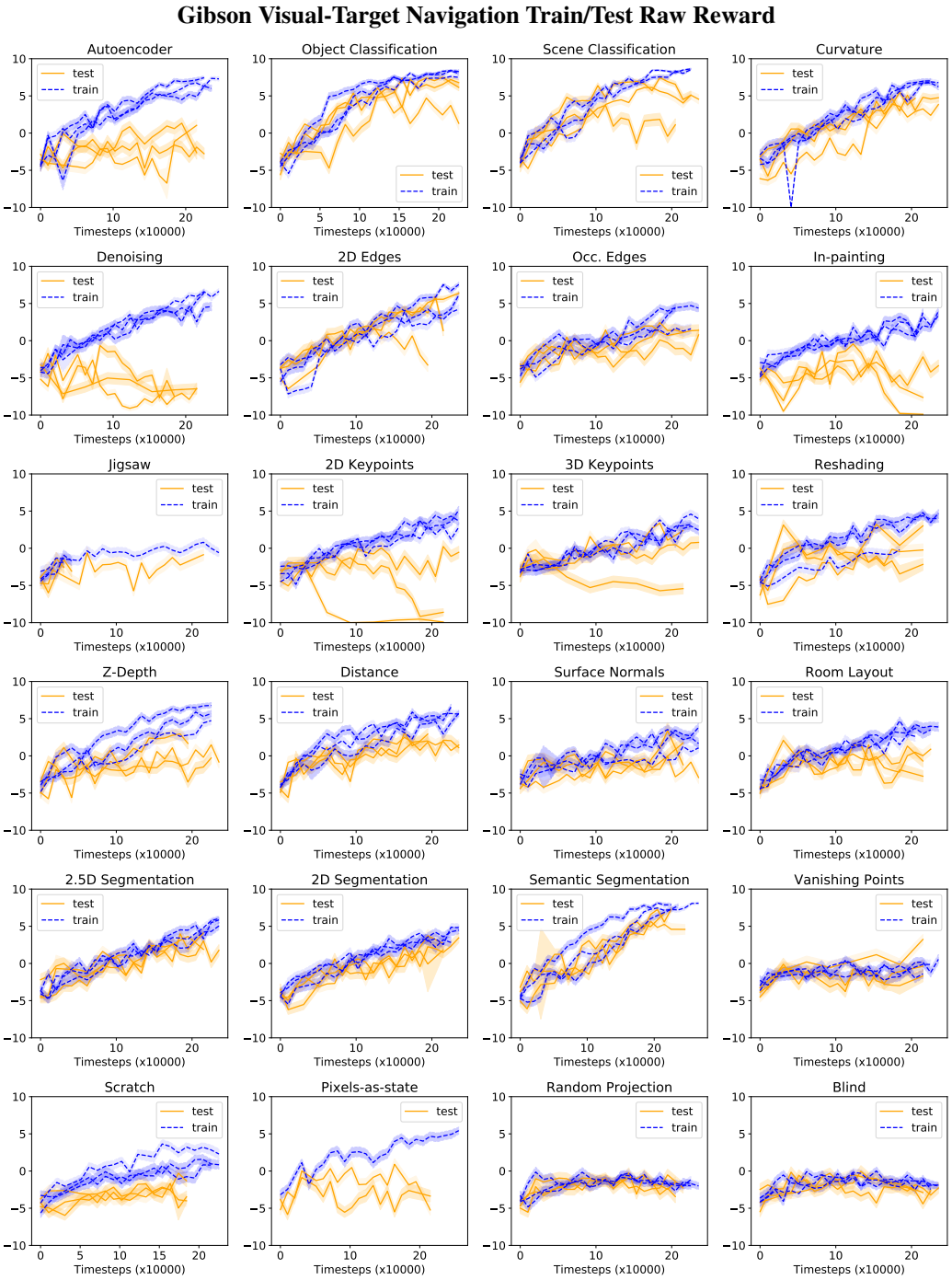

Figure 13: **Full Learning Curves (Gibson Navigation).** Full train and test curves for all features and baselines in the navigation task in the Gibson environment.

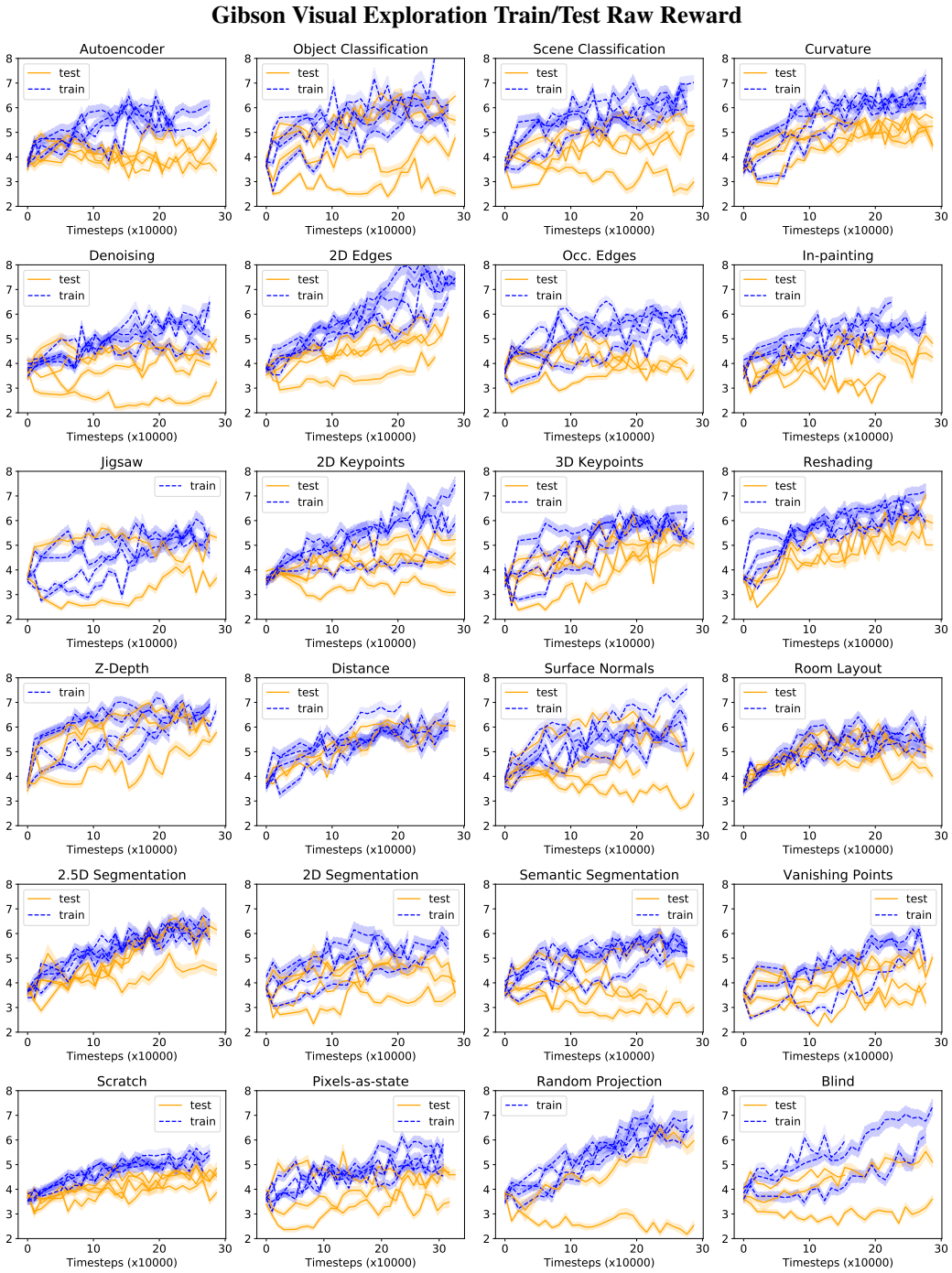

Figure 14: **Full Learning Curves (Gibson Exploration).** Full train and test curves for all features and baselines in the exploration task in the Gibson environment.

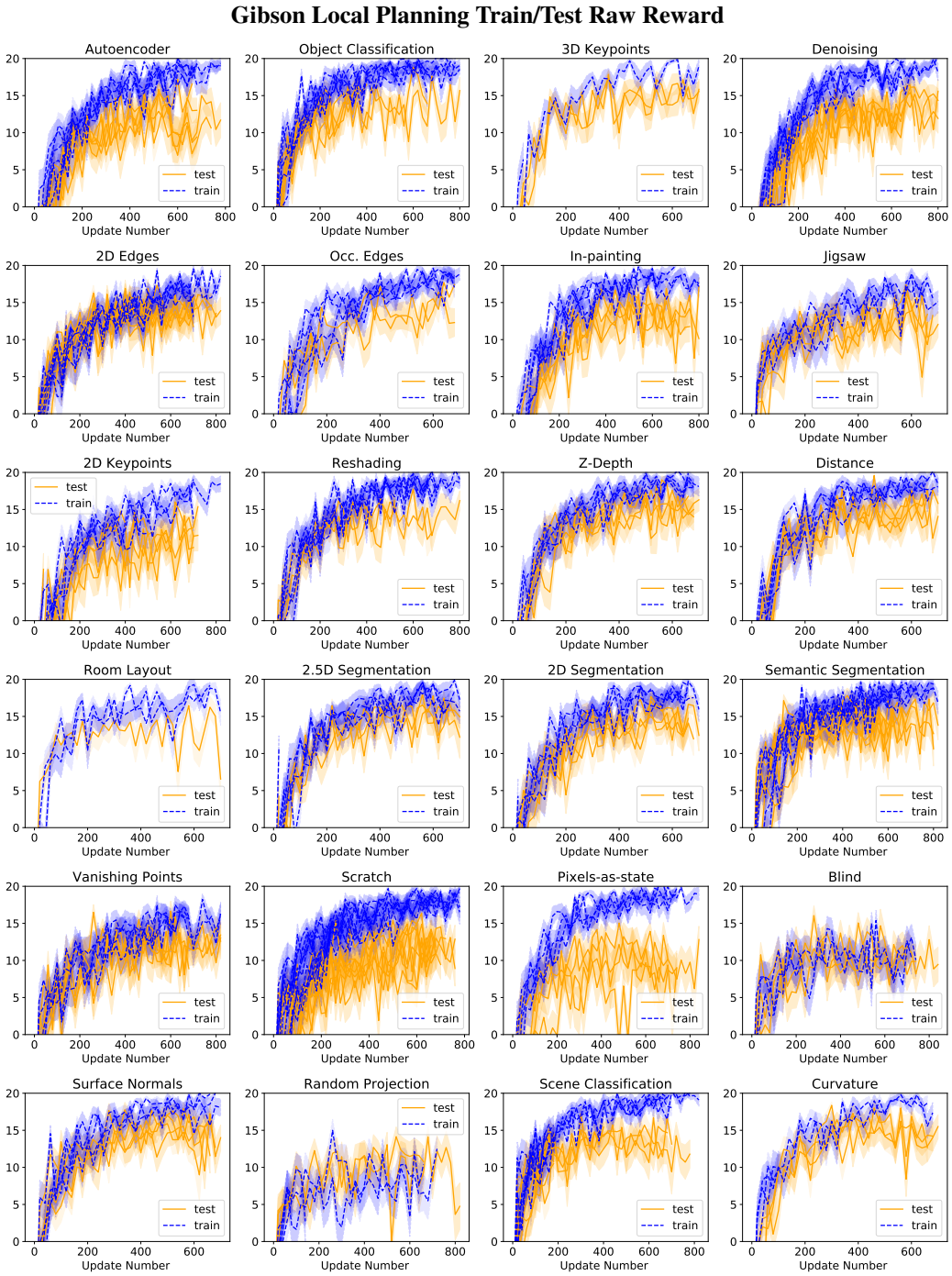

Figure 15: **Full Learning Curves (Gibson Local Planning).** Full train and test curves for all features and baselines in the planning task in the Gibson environment.

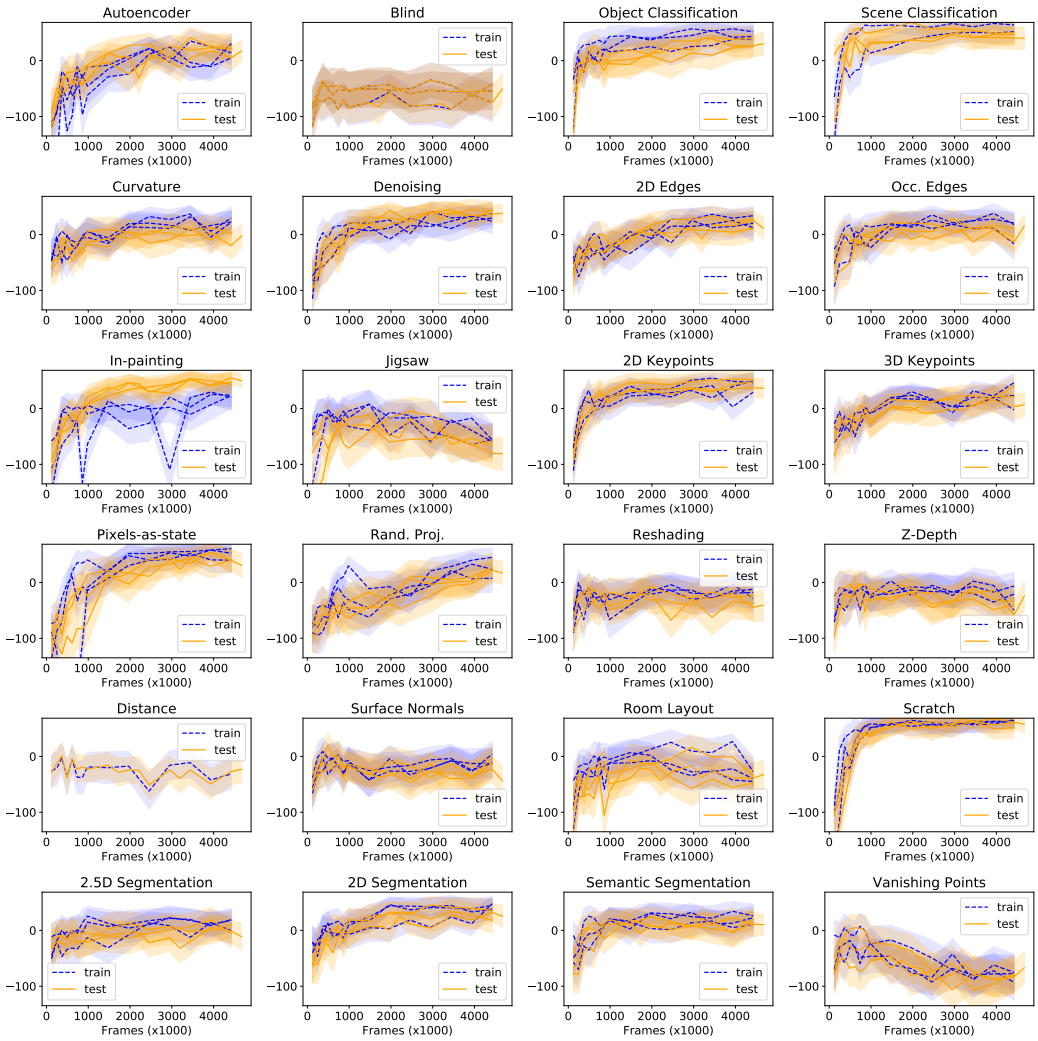

Figure 16: **Full Learning Curves (Doom Navigation).** Full train and test curves for all features and baselines in the navigation task in the ViZDoom environment.

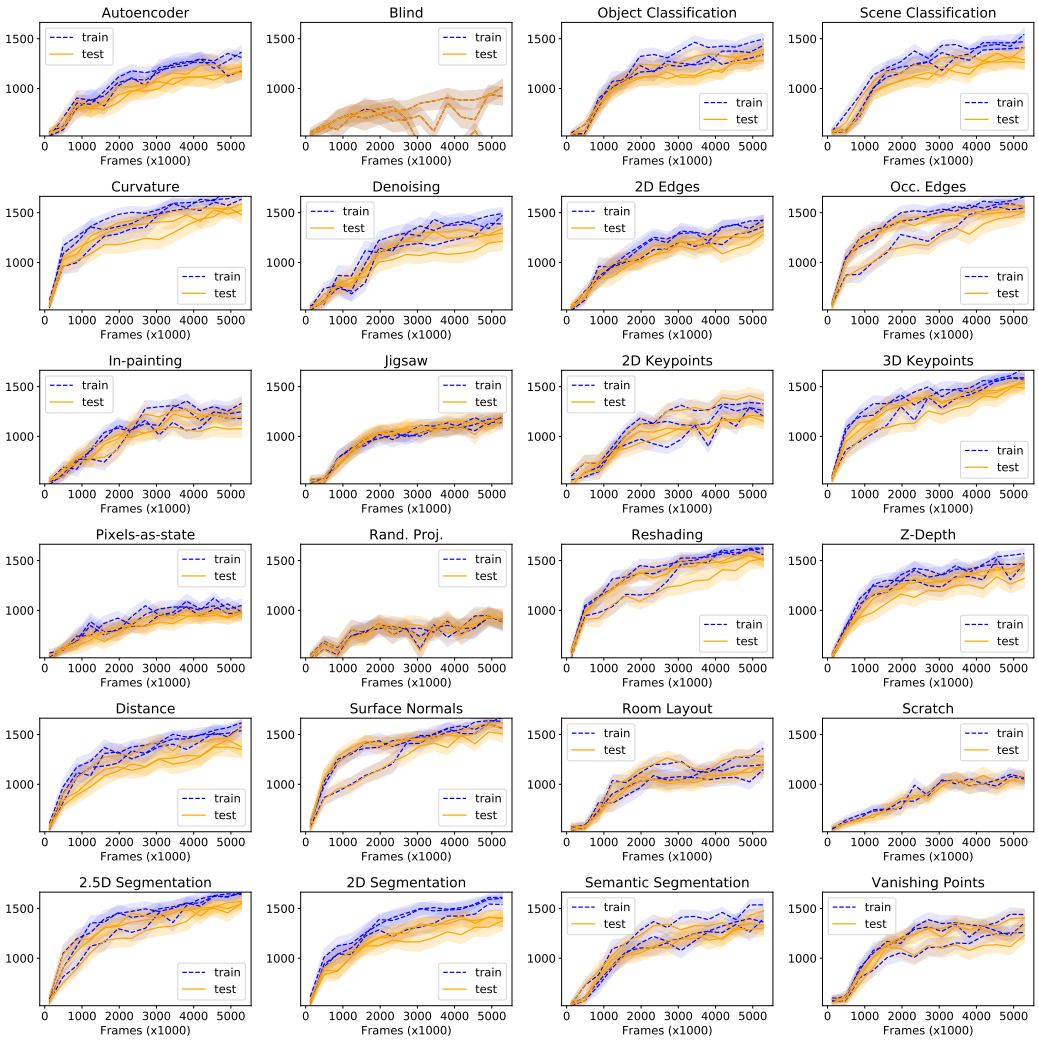

Figure 17: **Full Learning Curves (Doom Exploration).** Full train and test curves for all features and baselines in the exploration task in the ViZDoom environment.