Task Programming: Learning Data Efficient Behavior Representations Supplementary Material

Jennifer J. Sun[1] Ann Kennedy[2] Eric Zhan[1] David J. Anderson[1] Yisong Yue[1] Pietro Perona[1]
[1]Caltech [2]Northwestern University
Code & Project Website: https://sites.google.com/view/task-programming

We provide additional details and experimental results from task programming and TREBA.

- Section 1 describes the programs we use in the mouse and fly domain (Section 1.1) as well as experimental results with varying number of programs (Section 1.2).
- Section 2 provides implementation details on the representation learning architecture for TREBA and behavior classification models.
- Section 3 contains experimental results for decoder loss variations, time estimates, and classification samples.

1. Program Details and Experiments

1.1. Program Details

Programs for the Mouse Domain. We provide additional details on the programs listed in Table 1 in Section 3 of the paper. For the datasets in the mouse domain, programs are selected by domain experts based on the features used for mouse behavior classification in [4]. The experiments are recorded for a standard resident-intruder assay, where an intruder mouse is introduced to the cage of the resident mouse. Mouse 1 corresponds to the resident mouse and mouse 2 corresponds to the intruder mouse. These features are based on the anatomically defined keypoints tracked by the MARS tracker for each mouse: the nose, the ears, the base of the neck, the hips, and the base of the tail. A subset of the programs for the mouse domain is visualized in Figure 1, and all the programs we use for the mouse domain are listed below.

- Facing angle: Relative angle between orientation of the body of the mouse to the line connecting centroids of the two mice. The facing angle is computed for both mice. This describe how closely one mouse is facing the other mouse.
- Speed: Change in position of the centroid of the mouse across consecutive frames. The speed is computed for both mice. This property is especially important for helping identify aggressive behavior.
- Distance between nose of mouse 1 and 2: Distance between the nose keypoints of each mouse. Distance between noses can be used for identifying when the mice are interacting during social behavior such as sniff.
- Distance between nose of mouse 1 and tail of mouse 2: Distance between the nose keypoint of mouse 1 and base of tail keypoint of mouse 2. Distance between nose of mouse 1 and tail of mouse 2 can be used to identify when the mice are interacting during social behavior such as sniff.
- Head-Body Angle: For one mouse, the angle formed by the nose, neck, and base of tail keypoints. This angle is computed for each mouse. This attribute helps describe the body shape of the mouse, since it varies with changes to the relative orientation of the head and body of each mouse.
- Nose Movement: Nose movement of each mouse measured by speed relative to the movement of the centroid. This is computed for each mouse. This attribute describes the nose and head speed with respect to the center of the mouse, and can help identify aggressive behavior.

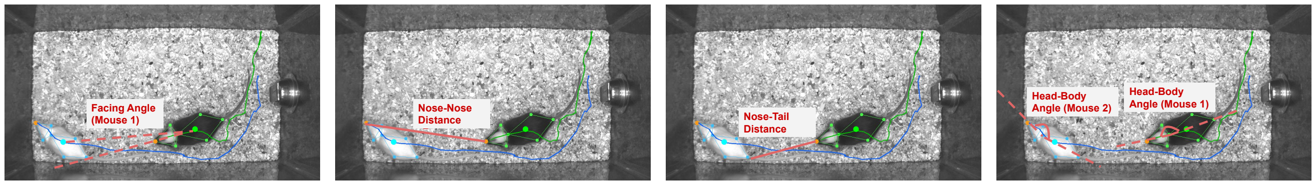

Figure 1. Visualizing behavior attributes for mouse dataset.

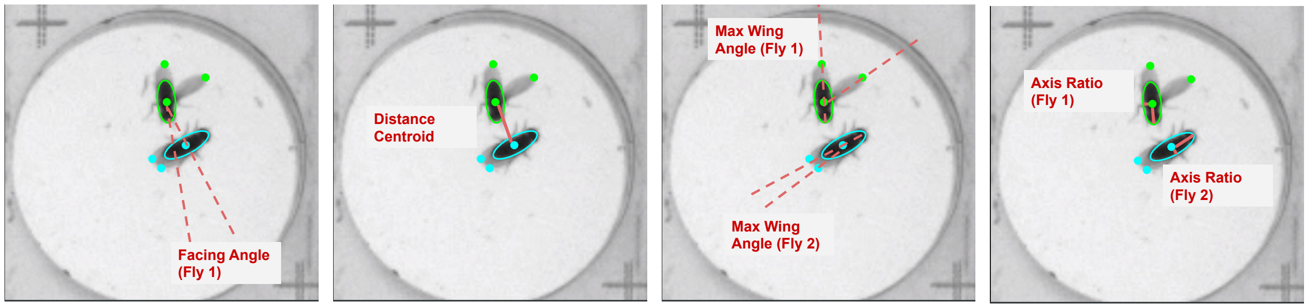

Figure 2. Visualizing behavior attributes for fly dataset.

Programs for the Fly Domain. For the datasets in the fly domain, these programs are selected by domain experts based on the features used for fly behavior classification in [3]. For each fly, the fly tracker tracks the centroid of the body with a fitted ellipse for the body, the left wingtip keypoint and the right wingtip keypoint. A subset of the programs is visualized in Figure 2 and all the programs we use for the fly domain are listed below.

- Angular speed: Change in heading direction of the fly across consecutive frames based on the fly body ellipse. The angular speed is computed for all flies. This attribute describes how fast the fly is turning and can help identify behaviors such as tussle and circle.
- Minimum and maximum wing angles: The wing angle is the angle between the wing tip keypoint, the centroid, and the point on the back of the body ellipse. Programs are used to compute both the minimum and maximum wing angles for each fly. The wing attributes are especially important behaviors defined by wing position and motion, such as wing extension and wing threat.
- Facing angle: Relative angle between orientation of the body of the fly to the line connecting centroids of the two flies. The facing angle is computed for both flies. The facing angle helps identify if a fly is facing in the direction of the other fly.
- Speed: Change in position of the centroid of the fly across consecutive frames. The speed is computed for all flies. This property helps identify behaviors such as lunge, which usually has high speed.
- Distance between centroid of fly 1 and 2: The distance between flies is often a good attribute to determine if the flies are interacting during social behavior.
- Ratio between the major and minor axis: The ratio between the major and minor axis length of the fly body ellipse. This is computed for each fly. This attribute is a useful description of the body shape of the fly. When the fly is tilting up, the ratio is usually smaller, and when the fly is flat against the surface, the ratio is usually larger.

1.2. Program Performance Results

We evaluate the representation learned using task programming for each individual program for the mouse and fly domains (Table 1 to Table 6). The evaluation procedure is the same as the main paper, where the MAP is averaged over nine runs with three random selection for each training fraction. MAP@$k\%$ corresponds to the classifier MAP supervised with $k\%$ of the training data. We note that average error reduction discussed in this section is computed across all evaluated training fractions from the main paper (1%, 2%, 5%, 10%, 25%, 50%, 100%).

MAP@k%	10	50	100
Domain-specific features	0.824	0.838	0.847
+ Head-Body Angle Mouse 1	0.853	0.868	0.874
+ Facing Angle Mouse 1	0.849	0.866	0.872
+ Distance Nose-Nose	0.849	0.866	0.868
+ Distance Nose-Tail	0.847	0.866	0.870
+ Speed Mouse 2	0.851	0.866	0.872
+ Head-Body Angle Mouse 2	0.846	0.866	0.872
+ Speed Mouse 1	0.847	0.862	0.874
+ Nose Movement Mouse 1	0.818	0.841	0.851
+ Facing Angle Mouse 2	0.814	0.834	0.843
+ Nose Movement Mouse 2	0.820	0.836	0.843
+ All Programs	0.853	0.868	0.877

Table 1. **Single Program Variations on MARS.** Average MAP of classifiers on MARS trained with features and with TREBA using the specified single program. The order of the programs are based on the average error reduction over all training fractions (highest error reduction at the top).

MAP@k%	10	50	100
Domain-specific features	0.824	0.838	0.847
+ 3 programs (A)	0.856	0.865	0.872
+ 3 programs (B)	0.850	0.866	0.872
+ 3 programs (C)	0.855	0.864	0.878

Table 2. **Additional Program Variations on MARS.** Average MAP of classifiers on MARS trained with features and with TREBA using the three programs. The programs are: (A) Nose Movement Mouse 1, Nose Movement Mouse 2, Facing Angle Mouse 2; (B) Facing Angle Mouse 1, Head-Body Angle Mouse 1, Head-Body Angle Mouse 2; (C) Speed Mouse 1, Nose Movement 1, Distance Nose-Nose.

MAP@k%	10	50	100
Domain-specific features	0.792	0.858	0.873
+ Distance Nose-Nose	0.811	0.876	0.889
+ Distance Nose-Tail	0.807	0.881	0.891
+ Speed Mouse 2	0.810	0.875	0.891
+ Facing Angle Mouse 1	0.811	0.879	0.890
+ Head-Body Angle Mouse 1	0.809	0.880	0.889
+ Speed Mouse 1	0.808	0.876	0.889
+ Head-Body Angle Mouse 2	0.802	0.870	0.881
+ Nose Movement Mouse 2	0.767	0.851	0.868
+ Nose Movement Mouse 1	0.765	0.852	0.869
+ Facing Angle Mouse 2	0.764	0.848	0.865
+ All Programs	0.808	0.876	0.888

Table 3. **Single Program Variations on CRIM13.** Average MAP of classifiers on CRIM13 trained with features and with TREBA using the specified single program. The order of the programs are based on the average error reduction over all training fractions (highest error reduction at the top).

MAP@k%	10	50	100
Domain-specific features	0.792	0.858	0.873
+ 3 programs (A)	0.811	0.879	0.889
+ 3 programs (B)	0.810	0.878	0.890
+ 3 programs (C)	0.807	0.877	0.887

Table 4. **Additional Program Variations on CRIM13.** Average MAP of classifiers on MARS trained with features and with TREBA using the three programs. The programs are: (A) Nose Movement Mouse 1, Nose Movement Mouse 2, Facing Angle Mouse 2; (B) Facing Angle Mouse 1, Head-Body Angle Mouse 1, Head-Body Angle Mouse 2; (C) Speed Mouse 1, Nose Movement 1, Distance Nose-Nose.

Mouse Program Evaluations. We train TREBA with each individual program from the mouse domain on Mouse100 using the consistency and contrastive losses, and evaluate the performance of domain-specific features+TREBA on MARS (Table 1) and CRIM13 (Table 3). In general, TREBA trained on a single program improves classifier performance, except for the bottom three programs and there is a high variation in performance (for example, Nose Movement Mouse 1 vs. Distance Nose-Nose). The two distance attributes, Head-Body Angle of Mouse 1, and Facing Angle of Mouse 1 generally performs the best across MARS and CRIM13 as a single program.

The best performing single program, Head-Body Angle Mouse 1 for MARS (Table 1) and Distance Nose-Nose for CRIM13 (Table 3), is comparable in performance to using all programs. When comparing average error reduction from baseline hand-designed features on MARS, TREBA using the top single program (Head-Body Angle Mouse 1, Table 1) achieves an error reduction of 14.5% and using all programs achieves an error reduction of 15.3%. On CRIM13, the top single program (Distance Nose-Nose, Table 3) achieves an error reduction of 9.3% and all programs achieves an error reduction of 9.5%. In contrast, the worst performing single program may reduce performance in the mouse domain. We study whether this performance variance can be reduced by adding more programs.

We experiment training TREBA using sets of three programs and evaluating behavior classification performance (MARS in Table 2, CRIM13 in Table 4). We note that program sets B and C are three randomly selected programs from the full mouse program list, and program set A consists of the worst performing three single programs. Despite program set A consisting of the lowest performing single programs, we see that for both MARS and CRIM13, the different sets of three programs are

MAP@$k\%$	10	50	100
Domain-specific features	0.774	0.829	0.868
+ Min. Wing Angle Fly 1	0.820	0.864	0.885
+ Speed Fly 1	0.818	0.856	0.878
+ Speed Fly 2	0.804	0.861	0.880
+ Angular Speed Fly 1	0.821	0.862	0.881
+ Max. Wing Angle Fly 2	0.814	0.859	0.882
+ Min. Wing Angle Fly 2	0.814	0.859	0.886
+ Distance Between Centroids	0.815	0.858	0.882
+ Facing Angle Fly 1	0.814	0.862	0.881
+ Axis Ratio Fly 1	0.809	0.855	0.882
+ Angular Speed Fly 2	0.811	0.853	0.879
+ Facing Angle Fly 2	0.811	0.853	0.876
+ Max. Wing Angle Fly 1	0.811	0.855	0.883
+ Axis Ratio Fly 2	0.797	0.852	0.880
+ All Programs	0.820	0.868	0.886

Table 5. **Single Program Variations on Fly.** Average MAP of classifiers on Fly trained with features and with TREBA using the specified single program. The order of the programs are based on the average error reduction over all training fractions (highest error reduction at the top).

MAP@$k\%$	10	50	100
Domain-specific features	0.774	0.829	0.868
+ 3 programs (A)	0.814	0.857	0.880
+ 3 programs (B)	0.814	0.857	0.878
+ 3 programs (C)	0.815	0.863	0.880
+ 7 programs (A)	0.819	0.869	0.889
+ 7 programs (B)	0.820	0.863	0.885
+ 7 programs (C)	0.815	0.860	0.882

Table 6. **Additional Program Variations on Fly**. Average MAP of classifiers on Fly trained with features and with TREBA using three and seven programs. The sets of three programs are: (A) Speed Fly 1, Min/Max Wing Angle Fly 1; (B) Speed Fly 2, Facing Angle Fly 1, Axis Ratio Fly 2; (C) Min Wing Angle Fly 2, Speed Fly 1, Axis Ratio Fly 1.
The sets of seven programs are: (A) Distance, Angular Speed Fly 1/2, Max Wing Angle Fly 1, Min/Max Wing Angle Fly 2, Facing Angle Fly 1; (B) Distance, Speed/Angular Speed Fly 1, Max Wing Angle Fly 1, Facing Angle Fly 2, Axis Ratio Fly 1/2; (C) Speed Fly 2, Min/Max Wing Angle Fly 1/2, Facing Angle Fly 1/2.

similar in performance and is comparable to using all programs (Table 2, Table 4). By training TREBA with three programs instead of one, the performance variation across program sets is much lower. We recommend that domain experts train with multiple programs, unless the best performing single program is known.

Fly Program Evaluations. We train TREBA on the Fly dataset without annotations using individual programs from the fly domain and evaluate on the Fly dataset (Table 5). Training with TREBA with any single expert-engineered program improves performance in the fly domain. We see that the speed and wing angle features generally perform the best. The top performing single program (Min. Wing Angle Fly 1, Table 5) achieves 19.4% average error reduction over baseline features, comparing to 21.2% using all programs. Similar to the mouse domain, if the best performing program is known ahead of time, we can achieve comparable performance to training TREBA using all programs. However, the best and worst single programs have a large variance, and we experiment with adding more programs.

We start by training on sets of three programs, and found that there is a gap in performance to using all programs (Table 6). We additionally experiment with sets of seven programs to close this performance gap. Training with randomly selected three or seven programs (Table 6) has much smaller variations across program selections compared to single programs (Table 5). For the fly domain, we found that training with seven programs is able to achieve comparable performance to using all programs.

2. Additional Implementation Details

We provide hyperparameters used in training TREBA (Table 7) and the classification models (Table 8). Our code is available at `https://github.com/neuroethology/TREBA`.

For training TREBA, the TVAE consists of a bi-directional GRU with 256 units for the encoder, followed by linear layers, with a latent dimension of 32. We take the encoding mean from the encoder, $\mathbf{z}_\mu$, as our learned representation of the trajectory. The decoder for the self-decoding task is also a GRU with 256 units followed by linear layers to predict the state in the next timestamp (by predicting the change from the current state to the next state). The decoder for the other tasks (attribute decoding, contrastive loss) consists of a fully connected neural network with 32 units. For the attribute consistency loss, we use the method proposed in [5] to train a 256 unit GRU to approximate non-differentiable programs. When the corresponding decoder is used, we weigh the consistency loss by 1.0, contrastive loss by 10.0, and the decoding loss by 1.0. We train TREBA using Adam optimizer with a learning rate of 0.0002. The input trajectories to the TREBA model are the detected keypoints for mouse and fly using the MARS tracker [4] and Fly tracker [3] respectively. At each frame, we stack the keypoints of the agents, and a trajectory in our experiment consists of 21 frames. We normalize the coordinates of pose

Dataset	Batch size	**z**-dim	Encoder Units	Decoder Units	Temperature t	Learning Rate
Mouse100	128	32	256	256	0.07	0.0002
Fly	128	32	256	256	0.07	0.0002

Table 7. **Hyperparameters for Representation Learning.**

Dataset	Batch size	Classifier Units (100%, 75%, 50%)	Classifier Units (25%, 10%)	Classifier Units (5%, 2%, 1%)	Learning Rate
MARS	512	256, 32	128, 16	64, 16	0.001
CRIM13	512	256, 32	128, 16	64, 16	0.001
Fly	512	256, 32	128, 16	64, 16	0.001

Table 8. **Hyperparameters for Classification Models.**

keypoints by the image pixel dimensions.

For classification, we use a shallow fully connected network with two hidden layers. We decrease the size of the network as the input training fraction decreases (Table 8). The model size and other hyperparameters are chosen based on the validation split. Our results are all reported on the test split. We train the classification models using cross-entropy loss and Adam optimizer with learning rate 0.001.

3. Additional Experimental Results

3.1. Decoder Loss Variations

We evaluate TREBA trained using different decoder losses on supervised behavior classification. The procedure is the same as described in the main paper. We evaluate performance given both our learned representation and one of either (1) raw keypoints or (2) domain-specific features designed by experts. The input keypoints to the classification model are the detected poses from the MARS tracker [4] and the Fly tracker [3]. The input domain-specific features are the hand-designed trajectory features for mouse [4] and fly [3]. The input features are a superset of the programs we use to train TREBA (listed in Table 1 from the main paper and described in Supplementary Material Section 1.1).

We compare the MAP of TREBA representations trained with different decoder losses in Table 9. The rows for TVAE and TVAE + Unsup. Contrast represents TREBA trained without programmed tasks and the remaining rows represents different combinations of decoder losses with programmed tasks. The average error reduction of these runs from baseline are shown in Table 2 in the main paper. Across all domains and training data amounts, we see that the learned representation improves classifier performance for both keypoints and domain-specific features. The improvements in performance are generally larger when we use keypoints, most likely because domain-specific features already contain informative features for classification. Furthermore, we experiment training TREBA without self-decoding, using unsupervised contrastive loss similar to [1, 2], and we see that the classifier MAP is lower than training with self-decoding using the TVAE loss.

Table 9 demonstrates that when using task programming, different decoder loss combinations (attribute consistency, decoding, and contrastive loss) are similar in performance in general, except when consistency loss is used alone. The lowest performing loss is when we use attribute consistency loss alone, which is applied to the generated trajectory and not directly to the representation. This result suggests that having at least one loss term directly applied on the representation (either decoding or contrastive loss) is beneficial.

Comparing task programming with TVAE loss alone, we see that TVAE loss is generally lower in performance (Table 9, Table 2 in the main paper). We note that TVAE loss alone corresponds to self-supervised learning with self-decoding only. Adding unsupervised contrastive loss to self-decoding improves performance relative to the TVAE, but we can improve the performance further using losses based on task programming (for example, TVAE+Contrastive+Consistency).

Random Program Inputs. We further experiment with training TREBA (Contrastive + Consistency), without expert-engineered programs, using a program that returns one of three classes randomly with equal probability for each trajectory. We found that the error reduction when using a program with random outputs is between training using TVAE alone and using unsupervised contrastive loss. On MARS relative to baseline features, TREBA with a random program achieves an error reduction of 14.0%, compared to 13.7% for TVAE, 14.3% for TVAE + Unsup. Contrastive, 15.3% for all programs (Table 2 in main paper). On Fly relative to baseline features, TREBA with a random program achieves an error reduction

Dataset	MARS			CRIM13			Fly		
MAP@$k\%$	10	50	100	10	50	100	10	50	100
Keypoints	0.588	0.635	0.656	0.538	0.621	0.648	0.348	0.519	0.586
+ TVAE	0.817	0.852	0.859	0.703	0.796	0.820	0.419	0.635	0.722
+ TVAE+Unsup. Contrast	0.815	0.852	0.866	0.706	0.813	0.837	0.521	0.667	0.739
+ TVAE+Consist	0.704	0.763	0.776	0.581	0.694	0.720	0.497	0.657	0.729
+ TVAE+Contrast	0.804	0.851	0.868	0.707	0.813	0.838	0.625	0.712	0.753
+ TVAE+Decode	0.825	0.857	0.872	0.719	0.828	0.848	0.666	0.737	0.773
+ TVAE+Contrast+Consist	0.822	0.856	0.866	0.722	0.821	0.837	0.650	0.707	0.750
+ TVAE+Decode+Consist	0.820	0.855	0.870	0.707	0.813	0.837	0.432	0.603	0.688
+ TVAE+Contrast+Decode	0.821	0.859	0.871	0.693	0.811	0.830	0.645	0.738	0.775
+ TVAE+Contrast+Decode+Consist	0.821	0.857	0.870	0.693	0.811	0.834	0.484	0.616	0.679
+ Unsup. Contrast	0.582	0.704	0.735	0.587	0.697	0.721	0.384	0.560	0.645
Domain-specific features	0.824	0.838	0.847	0.792	0.858	0.873	0.774	0.829	0.868
+ TVAE	0.850	0.866	0.869	0.808	0.874	0.885	0.791	0.852	0.880
+ TVAE+Unsup. Contrast	0.850	0.866	0.871	0.808	0.876	0.889	0.811	0.858	0.882
+ TVAE+Consist	0.824	0.841	0.853	0.775	0.854	0.871	0.812	0.856	0.882
+ TVAE+Contrast	0.853	0.868	0.872	0.811	0.876	0.889	0.834	0.869	0.888
+ TVAE+Decode	0.851	0.869	0.874	0.805	0.880	0.892	0.815	0.866	0.883
+ TVAE+Contrast+Consist	0.853	0.868	0.877	0.808	0.876	0.888	0.820	0.868	0.886
+ TVAE+Decode+Consist	0.848	0.868	0.873	0.808	0.872	0.888	0.781	0.838	0.862
+ TVAE+Contrast+Decode	0.846	0.867	0.876	0.811	0.878	0.892	0.810	0.862	0.885
+ TVAE+Contrast+Decode+Consist	0.851	0.866	0.872	0.813	0.879	0.892	0.783	0.842	0.868
+ Unsup. Contrast	0.830	0.847	0.853	0.787	0.858	0.874	0.784	0.842	0.866

Table 9. **Decoder Loss Variations.** Comparing data efficiency of TREBA trained with different decoder losses with respect to classifier MAP. Keypoints and domain-specific features represent baseline input features. TVAE represents training TREBA with self-decoding only and contrastive, decoding and consistency loss are described in Section 3.3 of the main paper.

of 13.6%, compared to 11.7% for TVAE, 16.1% for TVAE + Unsup. Contrastive, 21.2% for all programs (Table 2 in main paper). We tried adding more random programs during training, but did not observe an increase in performance. The lower performance of TREBA using random program inputs compared to all programs suggests that programs engineered using structured expert knowledge based on behavior attributes is important for improving the effectiveness of the learned representation.

3.2. Time Estimates

Based on domain expert estimates in neurobiology, behavior annotation takes 4 times the length of 30Hz videos, while task programming takes 5 to 10 minutes per program. We note that this time estimate is from domain experts familiar with data annotation and trajectory feature design. Applying this estimate to Figure 4 A2 B2 C2 in the main paper (for data efficiency with domain-specific features), in regions of low time investment ($<$ 2 hours), it is generally better for domain experts to annotate more data. For performance at $>$ 2 hours, task programming provides a better return on investment of the expert's time. Task programming requires an initial effort to produce the programs, which then scales to any data amount with no additional effort. Note that this time is variable depending on the domain expert and the domain, and our estimate is based on neurobiologists familiar with data annotation and trajectory feature design.

3.3. Classification Samples

We visualize classification samples for each dataset using input domain-specific features and TREBA (MARS in Figure 3, CRIM13 in Figure 4, Fly in Figure 5). We visualize the classifier using TREBA at the training fraction such that the classifier MAP matches that of the fully-supervised baseline feature performance. Comparing the samples qualitatively, we see that the classifier output with the full training set is comparable to TREBA with 10× reduced annotations on MARS and 2× reduced annotations on CRIM13 and Fly. We note that the classifier trained on reduced data alone (last row of Figures 3, 4, 5) is generally less accurate, compared to the classifiers trained with either full training data or with TREBA.

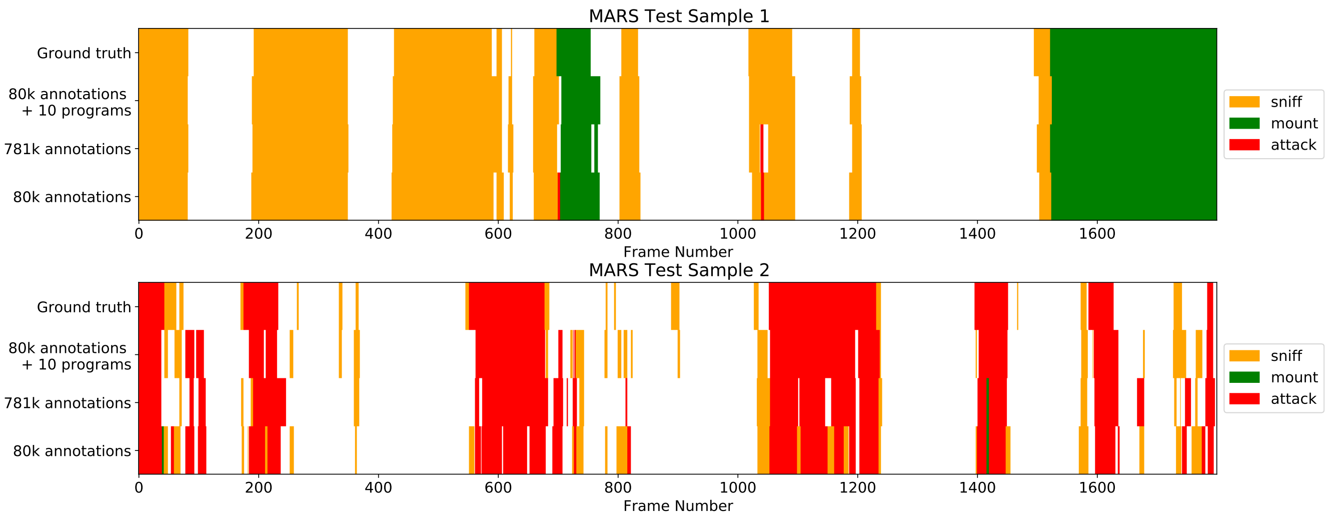

Figure 3. **Annotations on MARS Test Set.** Classifier annotations on MARS dataset vs. ground truth at 30Hz. For each sample, the second row corresponds to features + TREBA trained with 10 expert-engineered programs, with $\sim$ 10% of the supervised behavior annotations. The third row corresponds to training using domain-specific features on the full dataset. The last row corresponds to training using domain-specific features on $\sim$ 10% of data, without TREBA.

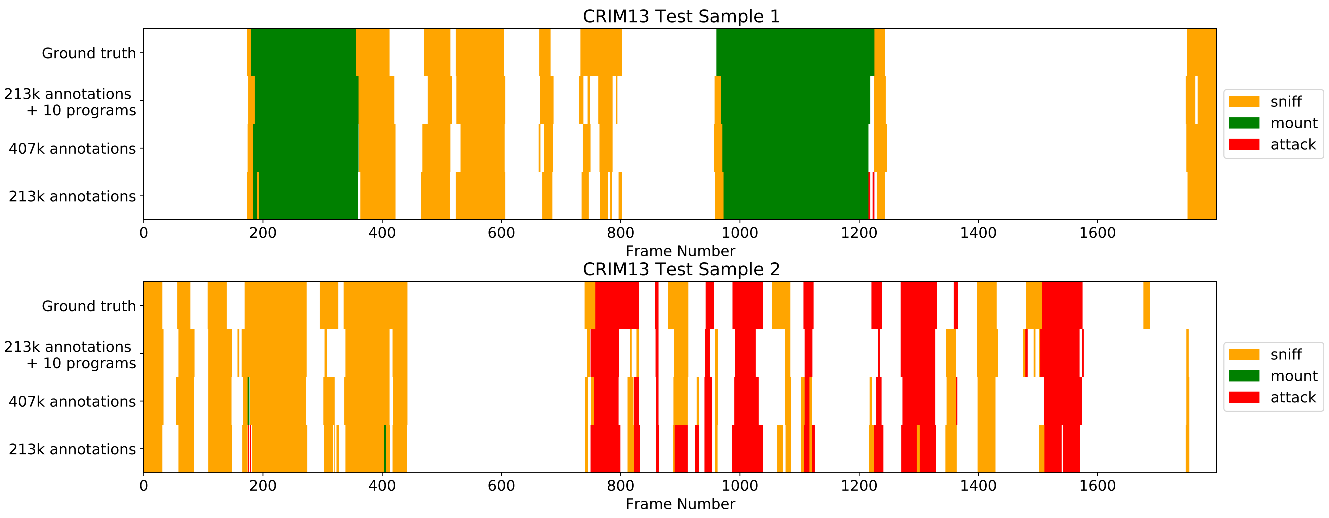

Figure 4. **Annotations on CRIM13 Test Set.** Classifier annotations on CRIM13 dataset vs. ground truth at 25Hz. For each sample, the second row corresponds to features + TREBA trained with 10 expert-engineered programs, with $\sim$ 50% of the supervised behavior annotations. The third row corresponds to training using domain-specific features on the full dataset. The last row corresponds to training using domain-specific features on $\sim$ 50% of data, without TREBA.

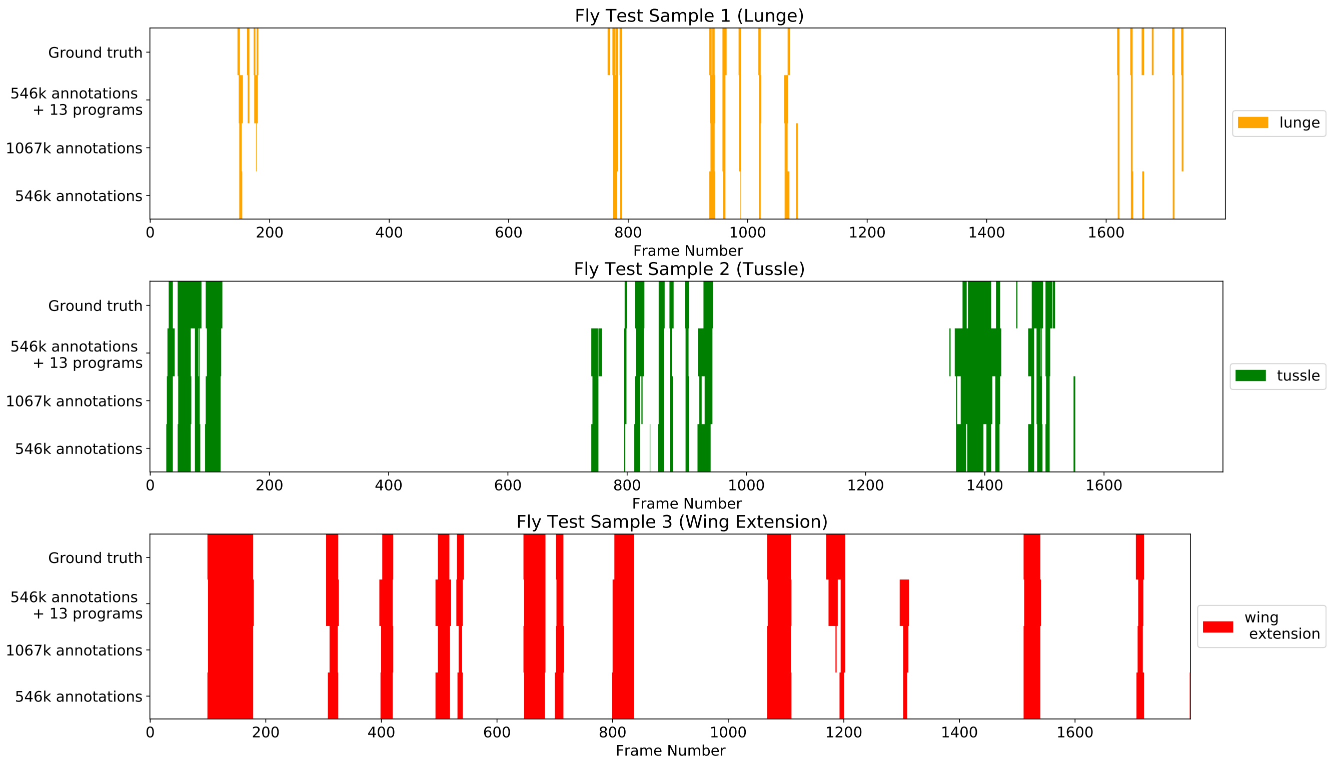

Figure 5. **Annotations on Fly Test Set.** Classifier annotations on Fly dataset vs. ground truth at 30Hz. For each sample, the second row corresponds to features + TREBA trained with 13 expert-engineered programs, with $\sim 50\%$ of the supervised behavior annotations. The third row corresponds to training using domain-specific features on the full dataset. The last row corresponds to training using domain-specific features on $\sim 50\%$ of data, without TREBA.

References

[1] Ting Chen, Simon Kornblith, Mohammad Norouzi, and Geoffrey Hinton. A simple framework for contrastive learning of visual representations. *ICML*, 2020. 5

[2] Ting Chen, Simon Kornblith, Kevin Swersky, Mohammad Norouzi, and Geoffrey Hinton. Big self-supervised models are strong semi-supervised learners. *arXiv preprint arXiv:2006.10029*, 2020. 5

[3] Eyrun Eyjolfsdottir, Steve Branson, Xavier P Burgos-Artizzu, Eric D Hoopfer, Jonathan Schor, David J Anderson, and Pietro Perona. Detecting social actions of fruit flies. In *European Conference on Computer Vision*, pages 772–787. Springer, 2014. 2, 4, 5

[4] Cristina Segalin, Jalani Williams, Tomomi Karigo, May Hui, Moriel Zelikowsky, Jennifer J. Sun, Pietro Perona, David J. Anderson, and Ann Kennedy. The mouse action recognition system (mars): a software pipeline for automated analysis of social behaviors in mice. *bioRxiv https://doi.org/10.1101/2020.07.26.222299*, 2020. 1, 4, 5

[5] Eric Zhan, Albert Tseng, Yisong Yue, Adith Swaminathan, and Matthew Hausknecht. Learning calibratable policies using programmatic style-consistency. *ICML*, 2020. 4