Homework 4

Out: Friday, April 28

Due: Friday, April 28 @ 11:59 pm EST

**Video links are provided in each section and are consolidated on the final page, in addition to url_main.txt*

***All eval commands have been adjusted to match the final announcement in EdStem (#1042, Mandi Zhao)*

2A. Why do we use *get_gaussian_scoremap* to generate the affordance target, instead of a one-hot pixel image?

--

Using a Gaussian scoremap to generate affordance targets has several advantages over one-hot pixel, such as smooth gradients, spatial invariance, robustness to noise, easier learning, and better localization. Gaussian scoremaps provide probabilistic representations of affordances, which allows the model to account for uncertainties and make more robust predictions. This is particularly useful in scenarios where there could be uncertainties or noise in the environment that can affect the robot's perception and decision making.

By comparison, one-hot encoding implies a deterministic grasp action, which may not be ideal as it assumes the robot is certain about specific grasp locations. Whereas Gaussian scoremaps can represent multiple affordances by having overlapping distributions that enable the model to learn various ways of interacting with the objects. This flexibility in representation helps the model predict and adapt to different grasp possibilities, which ultimately leads to more robust and adaptable models that are capable of making informed decisions in complex situations or environments.

In this problem, we use *get_guassian_scoremap* because each object has more than one affordance and one-hot encoding would limit model convergence.

2B. What transformations are done in *self.aug_pipeline*?

--

self.aug_pipeline is an augmentation pipeline that applies a series of random transformations to the input images. This augmentation process improves the model's ability to generalize unseen data by increasing the diversity of the training samples. The *iaa.Sometimes* function is used to apply transformations with a 70% probability, which means the specified augmentations will only be applied to 70% of the images while the remaining 30% are left unchanged.

The *iaa.Affine* function is responsible for applying affine transformations to the images, which are geometric transformations that preserve collinearity and relative differences between points. There are two specific affine transformations in *AugmentedDataset*, translation and rotation. The translation parameter defines the range of translation for the image in both the horizontal (x) and vertical (y) directions; images can be translated randomly between -20% and 20% of its width and height, which helps the model learn to recognize objects regardless of their position. The rotation parameter specifies the range of rotation angles for the image, which is calculated as *angle_delta*/2 where *angle_delta = 180/8*. This allows the image to be randomly rotated, which helps the model learn to recognize objects at different orientations.

These transformations help the model become more robust and generalize unseen data. More variations in the input images improves model adaptability and reduces the likelihood that the model will overfit to specific patterns or structures in the training data.

2D. Report your training loss and test loss.

Our *train.py max_epoch* defaults to 101 epochs for the Visual Affordance tasks in the assignment and 201 epochs for the Action Regression tasks in the assignment. I experimented with a variety of epochs and observed that the model converges between 30 and 50 epochs. The best model used in the *Affordance* or *Action Regression* tasks is reported in each section and reflected in each video at the start of the run.

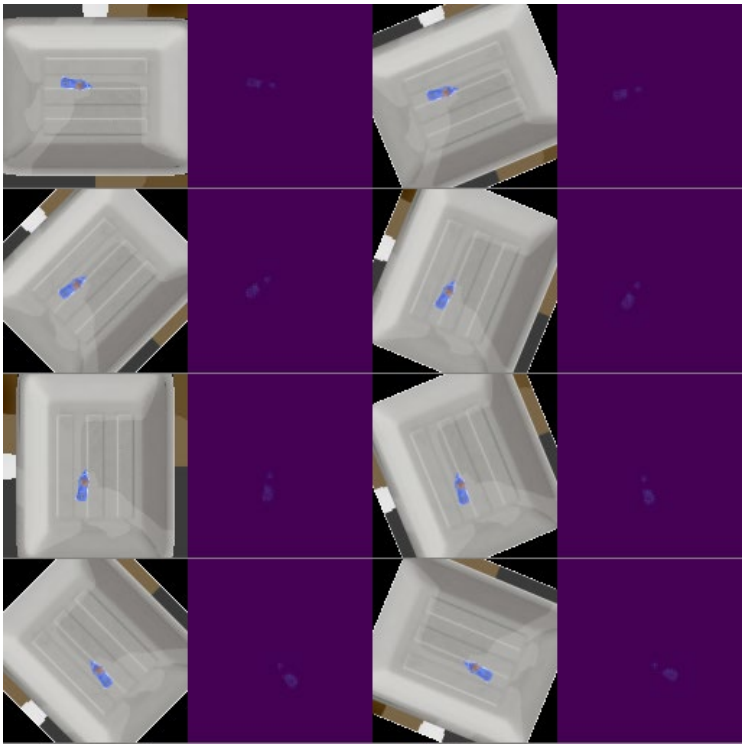
epoch 100, model_loss 0.0026762301567941904

Train loss: 0.0024
Test loss: 0.0033

2F. Report your success rate, provide a video link, include rotational images from *eval_pick_training_vis.*

```
python3 eval.py --model affordance --task pick_training
```

Testing on training objects. Success rate: 0.7333333492279053

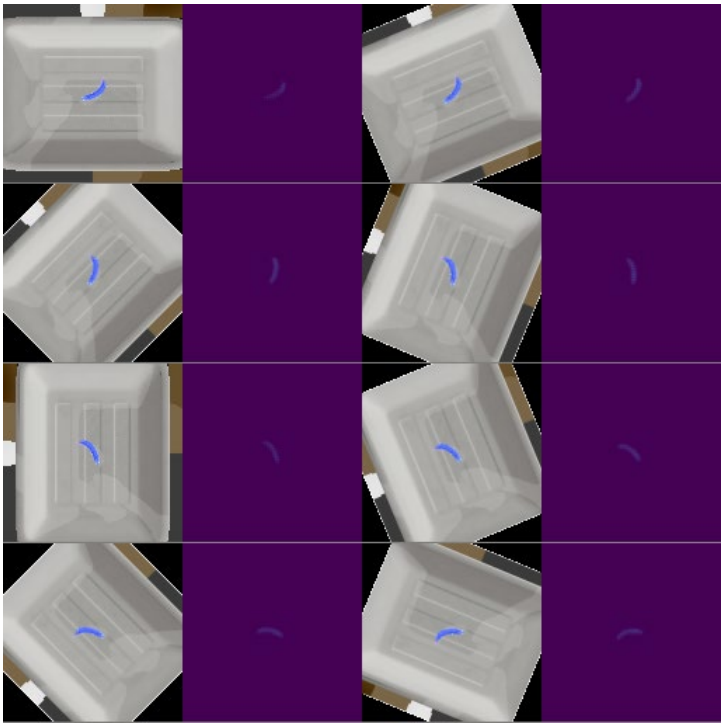


Video Link: HW4-2F

2G. Report your success rate, provide a video link, include rotational images from *eval_pick_testing_vis*.

```
python3 eval.py --model affordance --task pick_testing --n_past_actions 1
```

Testing on novel objects. Success rate: 0.5400000214576721



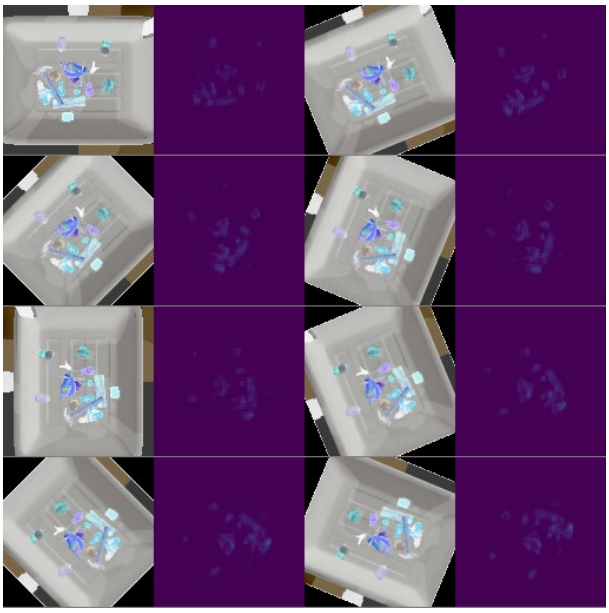
Video Link: HW4-2G

2H. Report the number of objects left in the bin, provide a video link.

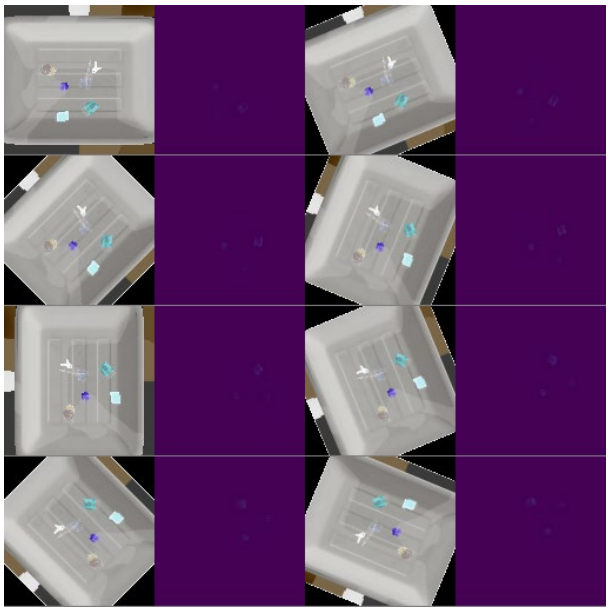
```
python3 eval.py --model affordance --task empty_bin --seed 2
```

9/15 objects moved, 6 objects left in the bin.

0.png



24.png

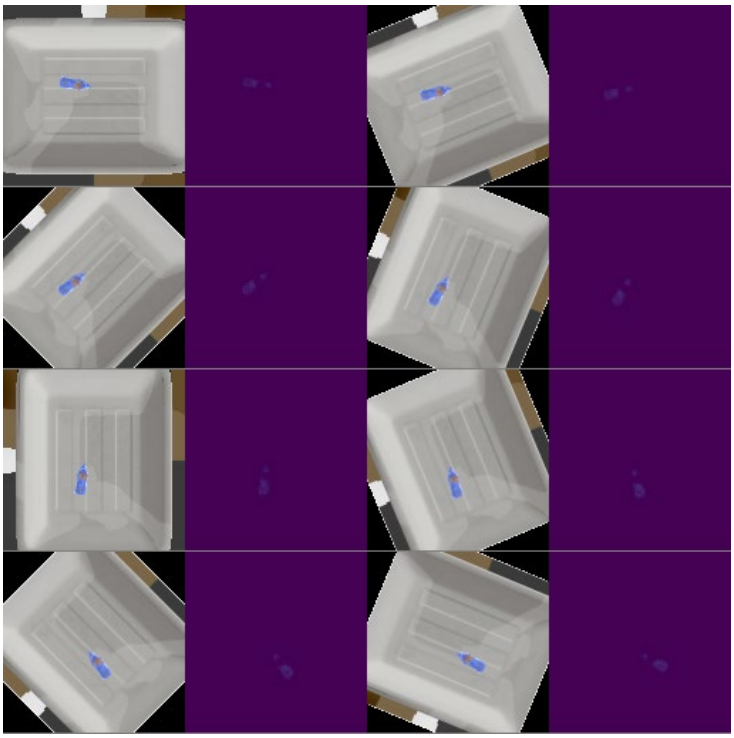


Video Link: HW4-2H

3A. Report success rate (not expected to improve), provide rotation images, record a video

```
python3 eval.py --model affordance --task pick_training
```

Testing on training objects. Success rate: 0.800000011920929

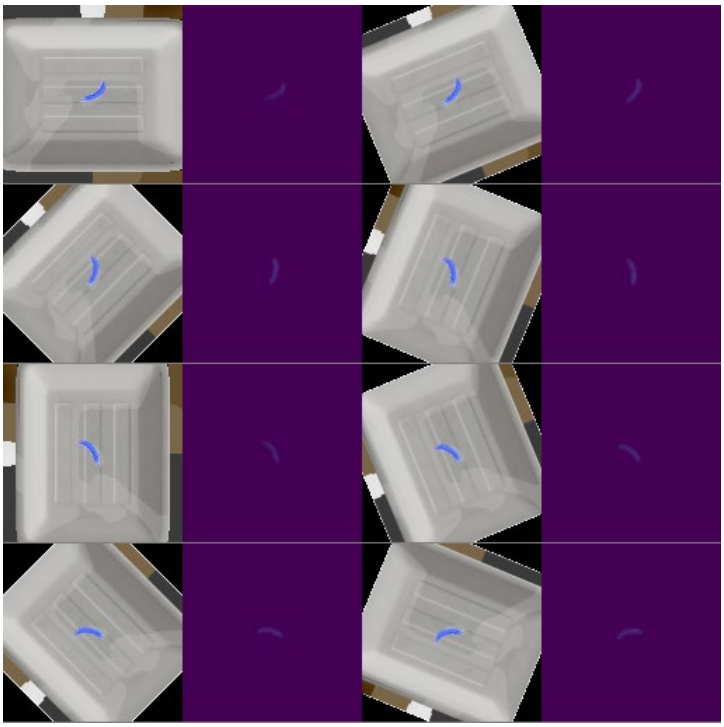


Video Link: HW4-3A

3B. Report success rate (expected to improve), provide rotation images, record a video

```
python3 eval.py --model affordance --task pick_testing --n_past_actions 8
```

Testing on novel objects. Success rate: 0.8500000238418579



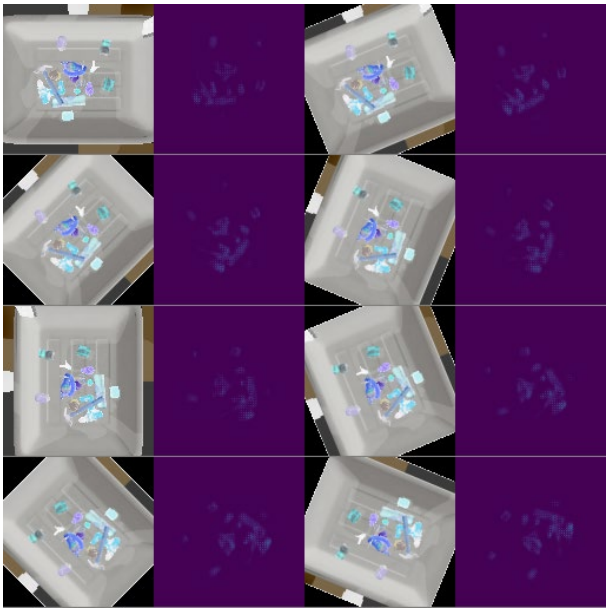
Video Link: HW4-3B

3C. Report success rate (expected to improve), provide rotation images, record a video

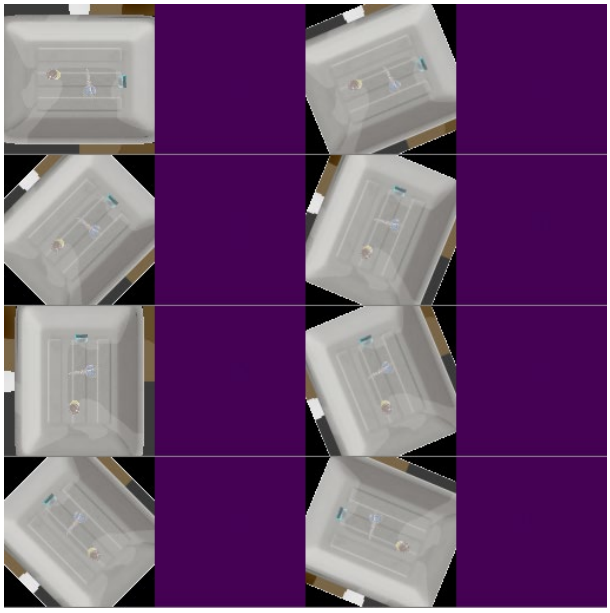
```
python3 eval.py --model affordance --task empty_bin --seed 2 --n_past_actions 25
```

12/15 objects moved, 3 objects left in the bin.

0.png



24.png



Video Link: HW4-3C

3D. Discuss how the performance from each evaluation run is different from Problem 2.

--

We should first acknowledge that only parts 3B and 3C are expected to improve on the performances observed in Problem 2, as a result of the inclusion of a past action buffer. Despite this, results did improve in 3A with an 80% success rate during testing on training objects. However, this bump from 73% in Problem 2F was not an evident performance improvement and there is some variability in the results of 2F that likewise matched or exceeded the 80% success rate observed in 3A.

Parts 3B and 3C were expected to demonstrate a material improvement over its counterparts in 2G and 2H, due to the inclusion of a past action buffer. In 3B, there was a noticeable 31% increase in accuracy. Likewise, there was a noticeable increase in objects successfully removed from the bins in 3C, where 12 of 15 objects (80%) were removed from the bin, compared to 9 of 15 objects (60%) removed in 2H.

Therefore, we can deduce that the inclusion of the past action buffer significantly increased the success rates of our visual affordance model.

4A. Report Training Loss and Test Loss, include image from *action_regression/training_vis*

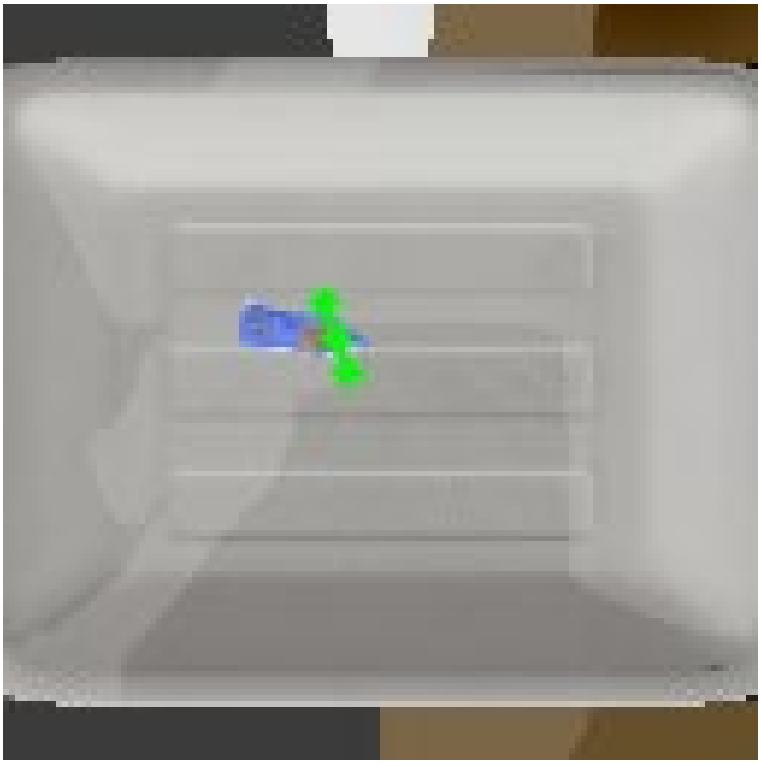
epoch 110, model_loss 0.07359454035758972

Train loss: 0.0098
Test loss: 0.0742

4B. Report success rate, include image of *YcbMustardBottle_2.png*, record a video

```
python3 eval.py --model action_regression --task pick_training
```

Testing on training objects. Success rate: 0.20000000298023224



Video Link: HW4-4B

4C. Report how many objects remain in the bin, record a video, explain why this method performs worse than Visual Affordance.

```
python3 eval.py --model action_regression --task empty_bin --seed 2
```

Our *train.py max_epoch* defaults to 201 epochs for the *Action Regression* training in this part of the assignment. I experimented with a variety of epochs and observed that the model converges well below 100 epochs. Training the model in roughly 110 epochs proved sufficient in parts 4A and 4B but was insufficient in 4C where 201 epochs resulted in more objects being removed from the bin. Specifically, the model trained on 110 epochs or less removed 00/15 objects from the bin, whereas the model trained on 201 epochs removed 03/15 objects from the bin.

Despite the difference in success rates, there was little indication that the hyperparameters influenced the outcome. Instead, success rates were primarily influenced by the initial placement of the objects in the bin. The best model used in the *Action Regression* tasks is reported in each section and reflected in each video at the start of the run. Performance videos have been provided for both models completing *task empty_bin,* the reported images and results below observe the performance of the model trained on 201 epochs.

03/15 objects moved, 12 objects left in the bin.

0.png 24.png

Video Link: HW4-4C (110E), HW4-4C (201E)

The action regression method performs worse than visual affordance because of the inability to effectively encode relationships between different aspects of the action, such as the coordinates and the angle, which can result in inaccurate predictions even when the loss function indicates otherwise. Additionally, action regression methods can be sensitive to minor prediction inaccuracies that can lead to poor performance. By comparison, visual affordance approaches can handle slight inaccuracies in coordinates and angles better as they focus on predicting the effect of an action on the environment.

Action regression methods also do not use Gaussian prediction to produce probabilistic output, which can negatively impact the model's ability to generalize and handle uncertainty. Furthermore, action regression methods are more prone to misclassification, especially when working with limited datasets that do not utilize visual affordance information to improve predictions. In general, visual affordance methods can better handle uncertainty and generalize new situations that ultimately lead to improved performance.

HW4 Video Recordings

https://drive.google.com/drive/folders/1yhYJWe1droKmkgnHXk7LjfzZ2IVX1Cp4?usp=share_link

HW4-2F

https://drive.google.com/file/d/1bf62jtZALwIFWGhjaSbHQrniqE6V1vgy/view?usp=share_link

HW4-2G

https://drive.google.com/file/d/1isff75hyWrZsrgrdq6lT5gnf5-yTh1Wp/view?usp=share_link

HW4-2H

https://drive.google.com/file/d/1GXUjw0WlHa1BGR4j5T0IFBtSOYa4t68f/view?usp=share_link

HW4-3A

https://drive.google.com/file/d/1KgNznzV0tdo4CzbYUT5n8sNh37F6rGEd/view?usp=share_link

HW4-3B

https://drive.google.com/file/d/1ZKUn8pEs7wHYU75xRS2eGD2ljzDZAtPr/view?usp=share_link

HW4-3C

https://drive.google.com/file/d/1Fk1ZmP-cnpzK2sH5yAeSlkVpOiqKMrak/view?usp=share_link

HW4-4B

https://drive.google.com/file/d/1wP_PyacGWfUsRl7rJ3ZthEfOHOzqsX7-/view?usp=share_link

HW4-4C (110E)

https://drive.google.com/file/d/1m7EO0VKlOD2Wy4Ia9bTbCEVP5qWbrppZ/view?usp=share_link

HW4-4C (201E)

https://drive.google.com/file/d/1N8g9j1j3tBb3aLzRlVYGYVtzTcr7sYi7/view?usp=share_link