Computational Aspects of Robotics
Professor Song
HW 4 Writeup
Alice Diakova
4/28/23

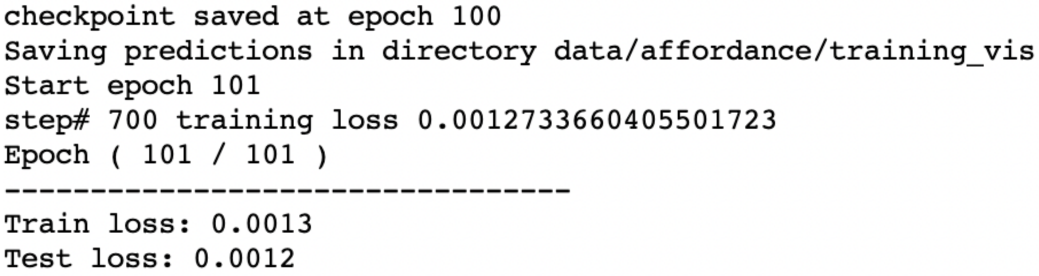
2a:

For affordance target maps, gaussian scoremaps have benefits like smoothness, noise resistance, handling of localization ambiguity, and improved training convergence. Contrary to one-hot pixel pictures, Gaussian scoremaps offer smooth, continuous scores, are less affected by noise, capture ambiguity, and offer flexibility in training with continuous gradients, leading to more precise and visually pleasing predictions.

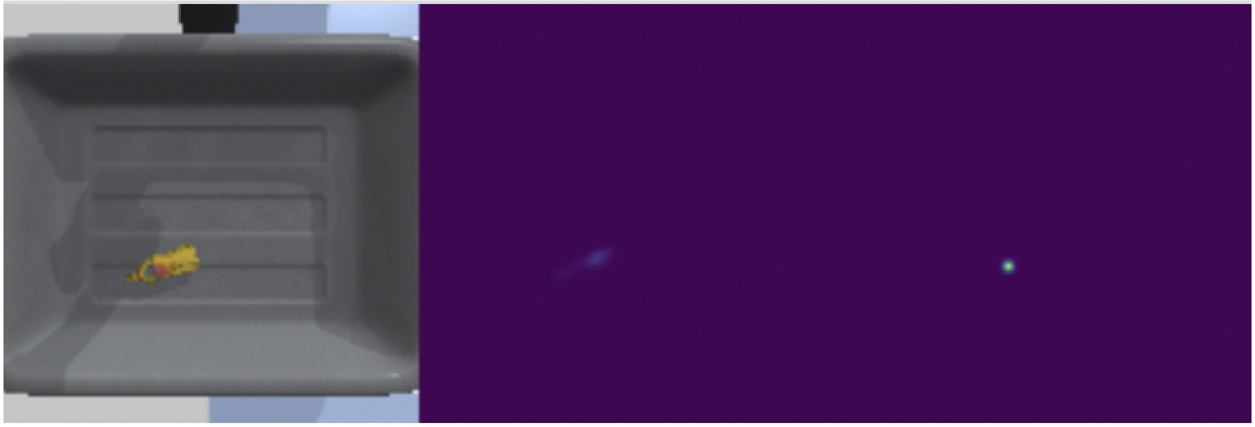
2b:

The transformations done in self.aug_pipeline are as follows: For 70% of the images, the images are translated by -20% to 20% on the x- and y-axis independently. Then, the images are rotated by -angle_delta/2 to angle_delta/2 degrees.

2d:



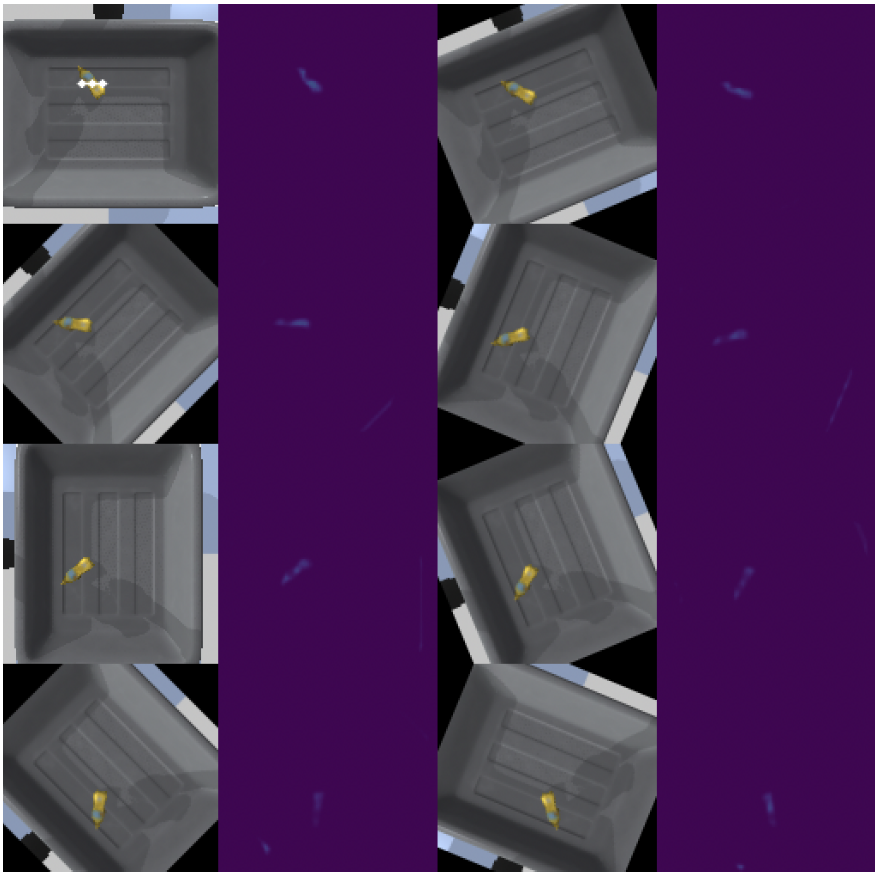
```
checkpoint saved at epoch 100
Saving predictions in directory data/affordance/training_vis
Start epoch 101
step# 700 training loss 0.0012733660405501723
Epoch ( 101 / 101 )
---------------------------------
Train loss: 0.0013
Test loss: 0.0012
```



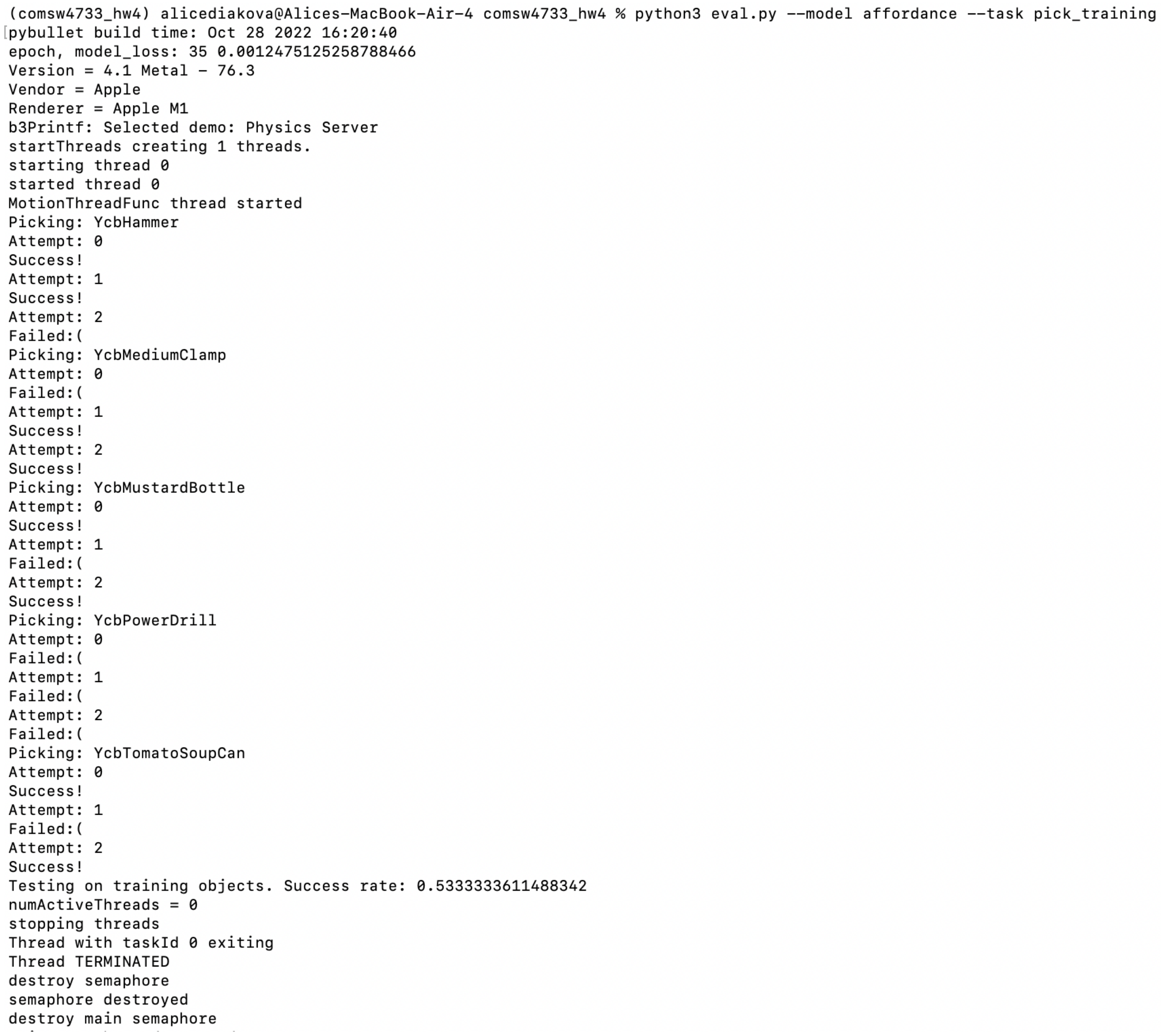
2f:

Success rate is 53.3%

The following is the image YcBMustardBottle_0.png:



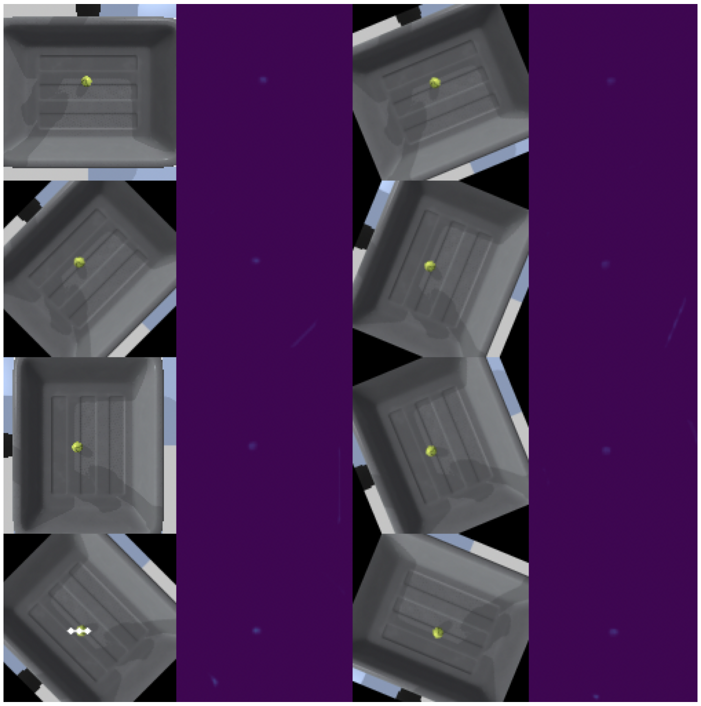
Note: The first time I ran the script for this part the program ran smoothly and displayed the output displayed in the image below. When I tried running it again to screen record it I kept running into an error involving “pybullet.error: getJointState failed”. I could not figure out how to solve this problem so the recording submitted for part 2f shows the error instead of successful operation, unfortunately.



```
(comsw4733_hw4) alicediakova@Alices-MacBook-Air-4 comsw4733_hw4 % python3 eval.py --model affordance --task pick_training
[pybullet build time: Oct 28 2022 16:20:40
epoch, model_loss: 35 0.0012475125258788466
Version = 4.1 Metal - 76.3
Vendor = Apple
Renderer = Apple M1
b3Printf: Selected demo: Physics Server
startThreads creating 1 threads.
starting thread 0
started thread 0
MotionThreadFunc thread started
Picking: YcbHammer
Attempt: 0
Success!
Attempt: 1
Success!
Attempt: 2
Failed:(
Picking: YcbMediumClamp
Attempt: 0
Failed:(
Attempt: 1
Success!
Attempt: 2
Success!
Picking: YcbMustardBottle
Attempt: 0
Success!
Attempt: 1
Failed:(
Attempt: 2
Success!
Picking: YcbPowerDrill
Attempt: 0
Failed:(
Attempt: 1
Failed:(
Attempt: 2
Failed:(
Picking: YcbTomatoSoupCan
Attempt: 0
Success!
Attempt: 1
Failed:(
Attempt: 2
Success!
Testing on training objects. Success rate: 0.5333333611488342
numActiveThreads = 0
stopping threads
Thread with taskId 0 exiting
Thread TERMINATED
destroy semaphore
semaphore destroyed
destroy main semaphore
```

2g: The success rate is 31% and the following testing image is YcBTennisBall_9.png

- The image is similar to those of the previous part in that the affordance maps seem to be the same intensities and there tends to be only one cluster of points in the map per object. One way the testing images differ from the training images are that many of the testing affordance maps have the robot itself lighting up in a way that makes the system decide to grasp the object at the robots location so it grasps itself. This phenomenon can be seen in the image below where the affordance map of the lower-left-most image pair contains two distinct points, one being the object and the other being the robot. This problem is what caused a majority of the failures in this part.

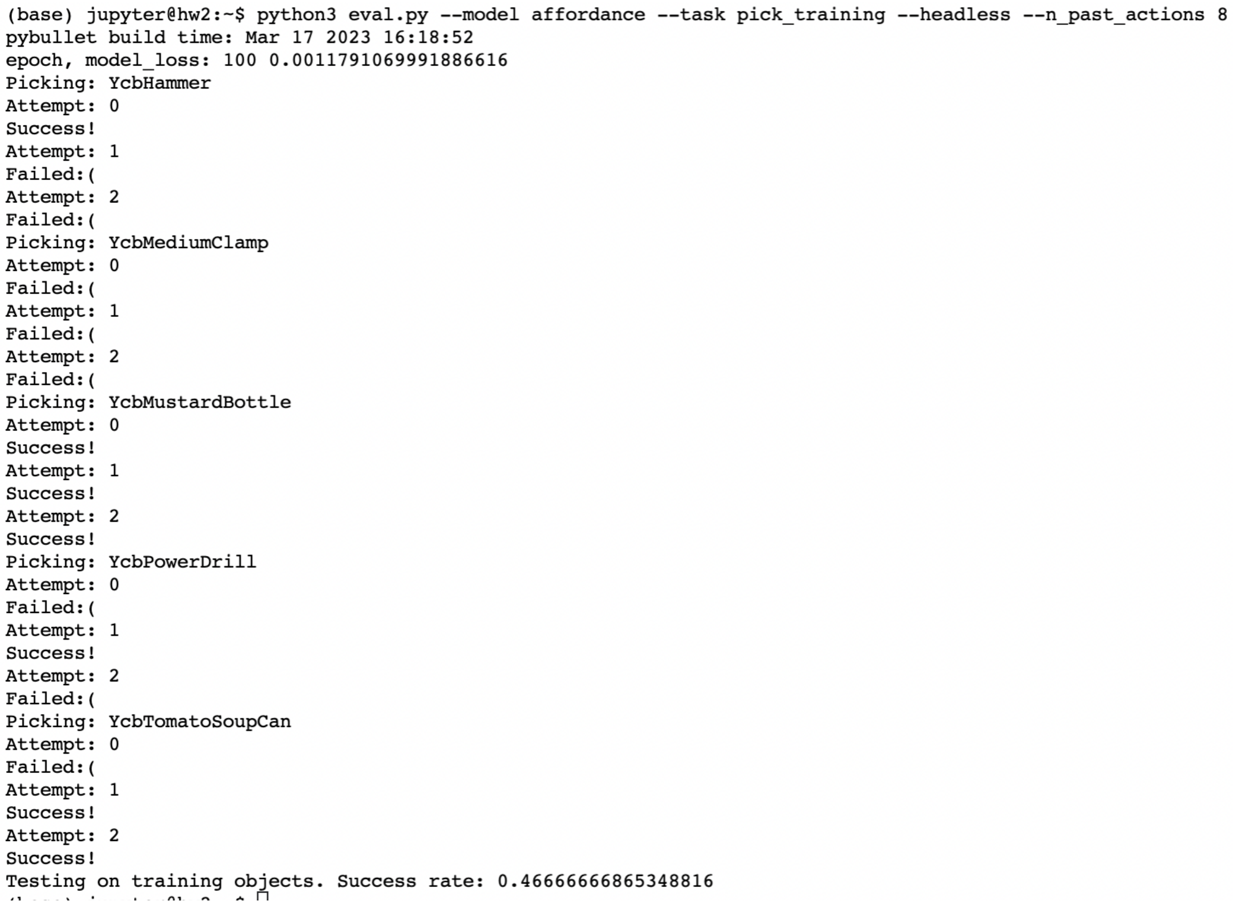


2h: Since I was having trouble with the “pybullet.error: getJointState failed” error, the screen recording for this part only includes the terminal and not the Pybullet simulation.
After running the script for this part, there were 7/15 objects picked up, and 8 objects still left.

2i:
The method of visual affordance focuses on learning the relationship between an object's visual affordance and its potential actions rather than learning to recognize items only based on their appearance. By doing this, the model can develop the ability to identify items based on their affordances, which are much more insightful than merely their visual appearance. This information can help the model generalize to new objects that it has not seen before, as long as they have similar affordances to the ones it has seen.

Problem 3 (for all parts of problem 3 I was having trouble generating the Pybullet simulation due to the same getJointState so I only have screenshots of the terminal performances):

3a:



```
(base) jupyter@hw2:~$ python3 eval.py --model affordance --task pick_training --headless --n_past_actions 8
pybullet build time: Mar 17 2023 16:18:52
epoch, model_loss: 100 0.0011791069991886616
Picking: YcbHammer
Attempt: 0
Success!
Attempt: 1
Failed:(
Attempt: 2
Failed:(
Picking: YcbMediumClamp
Attempt: 0
Failed:(
Attempt: 1
Failed:(
Attempt: 2
Failed:(
Picking: YcbMustardBottle
Attempt: 0
Success!
Attempt: 1
Success!
Attempt: 2
Success!
Picking: YcbPowerDrill
Attempt: 0
Failed:(
Attempt: 1
Success!
Attempt: 2
Failed:(
Picking: YcbTomatoSoupCan
Attempt: 0
Failed:(
Attempt: 1
Success!
Attempt: 2
Success!
Testing on training objects. Success rate: 0.46666666865348816
(base) jupyter@hw2:~$ 
```

3d: The performance of each evaluation run was better than the evaluation runs in Problem 2. Each attempt clearly involves multiple tries if failures are consistently seen. In 3a the performance was roughly the same as it was in problem 2. In 3b the performance improved to 0.52 and in 3c the performance improved even more to 0.54.