Homework 2 Advanced BC Exploration Report
CS-665
Jared Miller
1/23/2026

For my exploration I decided on option 2 and chose the Action Chunking Transformer paper. I decided on option 2 because I didn't have any robotic equipment. I chose the paper I chose because I don't have much experience with Transformers and I wanted to learn more. I did my work in blocks of 1hr and 30m and documented my process here.

Work Block 1: To start this workblock off I knew that I needed a little more information on my problem. I started this research block with the following video:
Transformers, the tech behind LLMs | Deep Learning Chapter 5 and
Attention in transformers, step-by-step | Deep Learning Chapter 6 detailed notes on both videos are shown at the end of the report. After I understood more about transformers I started looking into what open source code was available for action chunking transformers. Even though I didn't end up using transformers in my end implementation the knowledge was still super valuable.

Work Block 2 and 3: I combined the report section for these two workblocks because I didn't take a break in between. To start this workblock I wanted to figure out exactly what implementation I would be doing. After some research I decided on comparing regular behavioral cloning to action chunking behavioral cloning. Unlike the paper my implementation won't include image processing and transformers because I am working on a simple problem. In my exploration the main idea I wanted to explore was how much of a difference action chunking made on overall performance.
I settled on the Pendulum problem from gymnasium to compare the results. I created multiple classes that were helpful here.

- bc_model.py: This file contains both the BCPolicy network and the ChunkedBCPolicy network. They are both nn.Module. This is where the neural networks are defined.
- bc_utils.py: This file contains both the Dataset classes to help with data handling.
- collect_demos: This file contains the logic to collect demos from the expert after the expert has already been trained.
- eval_bc.py: This file contains the logic to visualize the regular behavior cloning performance.
- eval_chunked_bc.py: This file contains the logic to visualize the action chunking behavioral cloning performance.
- evaluate_policy.py: This file contains the logic to qualitatively compare the two methods.
- test_expert.py: This file contains simple logic to ensure that the trained agent is good at the task.
- test.py: This file contains simple logic to ensure that the environment is working as expected.
- train_bc.py: This file contains the logic to train the regular behavior cloning policy

- train_chunked_bc.py: This file contains the logic to train the action chunking behavioral cloning policy.
  train_expert.py: This file contains the logic to train the expert using PPO

1. What did you learn about the algorithm/implementation that you didn’t know when you started this?

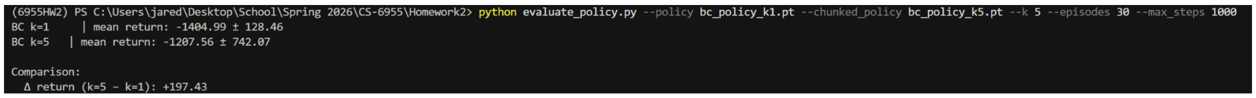
   I assumed that I was going to have to collect expert demonstrations myself. I had no idea there were so many libraries that already had policy learners that were generalizable. I ended up using PPO which was shockingly good. I was also surprised at how intuitive gym / gymnasium was to use.

2. What evidence do you have that your code/algorithm worked or failed?

   I compared regular behavioral cloning to action chunking behavioral cloning to see how much covariant shift caused a problem. If you run:

   python evaluate_policy.py --policy bc_policy_k1.pt --chunked_policy bc_policy_k5.pt --k 5 --episodes 30 --max_steps 1000

   you get the following output:



```
(6955HW2) PS C:\Users\jared\Desktop\School\Spring 2026\CS-6955\Homework2> python evaluate_policy.py --policy bc_policy_k1.pt --chunked_policy bc_policy_k5.pt --k 5 --episodes 30 --max_steps 1000
BC k=1     | mean return: -1404.99 ± 128.46
BC k=5   | mean return: -1207.56 ± 742.07

Comparison:
  Δ return (k=5 - k=1): +197.43
```

   This shows that even on a simple problem like the pendulum problem action chunking makes a big difference. In this case with no other changes besides an action chunking factor of 5 we got a +197.43 improvement. This was without any hyper-parameter tuning which was incredibly impressive.

3. What were some of the biggest challenges you faced?

   If I’m being honest the assignment was much easier and much more valuable than I expected. It was really hard to get started because I was very intimidated by the assignment but once I got going, things were super smooth sailing.

4. How successful were you at getting things to work?

   Very successful! I got everything working how I expected and also got the results that I expected.

5. In hindsight, what might you have done differently if you were to go back in time and redo this assignment?

   If I could go back in time I don’t know if there is anything I would do different. I didn’t hit any roadblocks and I felt like I learned a ton on this project.

6. How was AI helpful and/or frustrating in completing this assignment?

   Absurdly valuable. It helped me figure out how to use gymnasium, gave me ideas on what problem I should compare results with, and helped me get everything working.