Ira A. Fulton Schools of Engineering
Arizona State University

CSE 571: Artificial Intelligence

Neural Network for Collision Prediction Project

Purpose

The purpose of this project is to familiarize you with neural networks and their applications. More specifically, you will learn to collect training data for a robotics task and, in turn, design a neural network that can process this data. In the project, your goal is to help a small robot navigate a simulated environment without any collisions. To accomplish this, you will need to train a neural network using backpropagation that predicts whether the robot is going to collide with any walls or objects in the next time step. All of the theories and best practices introduced in the class are applicable here. Your task is to make sure the little fellow can safely explore its environment!

Objectives

Learners will be able to:

- Collect and manage a dataset used to train and test a neural network.
- Define and use PyTorch DataLoaders to manage PyTorch Datasets.
- Design your own neural network architecture in PyTorch.
- Evaluate and improve a neural network model and verify an application in simulation.

Technology Requirements

- System designed for use with Ubuntu 22.04
- Python and its related libraries. Using Anaconda is recommended
- Python 3.12
- Python libraries:
    - cython==3.0.311matplotlib==3.9.3
    - scikit-learn==1.5.2scipy==1.14.1
    - pymunk==6.9.0pygame==2.6.1
    - pillow==11.0.0
    - numpy==2.1.3noise==1.2.2

- torch==2.5.1
- torchvision==0.20.1

Directions

Part 1

Download “CSE 571_Part 1_Neural Network for Collision Prediction_Project Files.zip” from the “**Project Overviews and Resources**” page in the *Welcome and Start Here* module, carefully review the directions, and then start working in the provided starting code:

- Collect_data.py

The first task is to collect data that can be used to train the model. Please review “Image 1: PyGame Display” showing how the robot should wander around with no regard for its environment or avoiding collisions.

Collect a single sample per action containing, in order:

1. The five (5) distance sensor readings
2. The action
3. Whether or not a collision occurred (0: no collision, 1: collision)

Your data should be saved as a .csv file with seven (7) untitled columns. Please review “Image 2: Sample Learner Submission.CSV” of what your submission.csv should look like.

For grading purposes, submit your ‘submission.csv’ containing 100 data samples. For training in the future parts, you will need to collect much more than this.

Image 1.0: Part 1 PyGame Display

Image 1.0: PyGame Display shows a sample environment and the robot’s potential actions. The robot should wander around with no regard for its environment or avoiding collisions. You may want to enlarge the image to see it clearly.

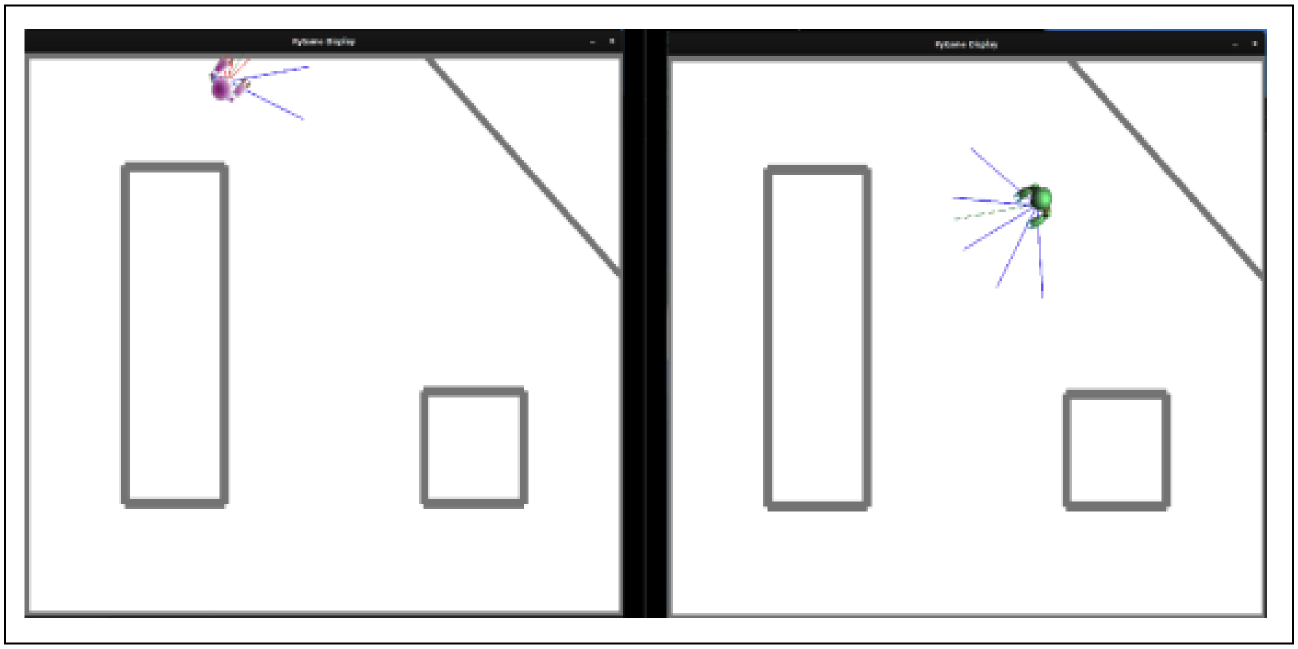
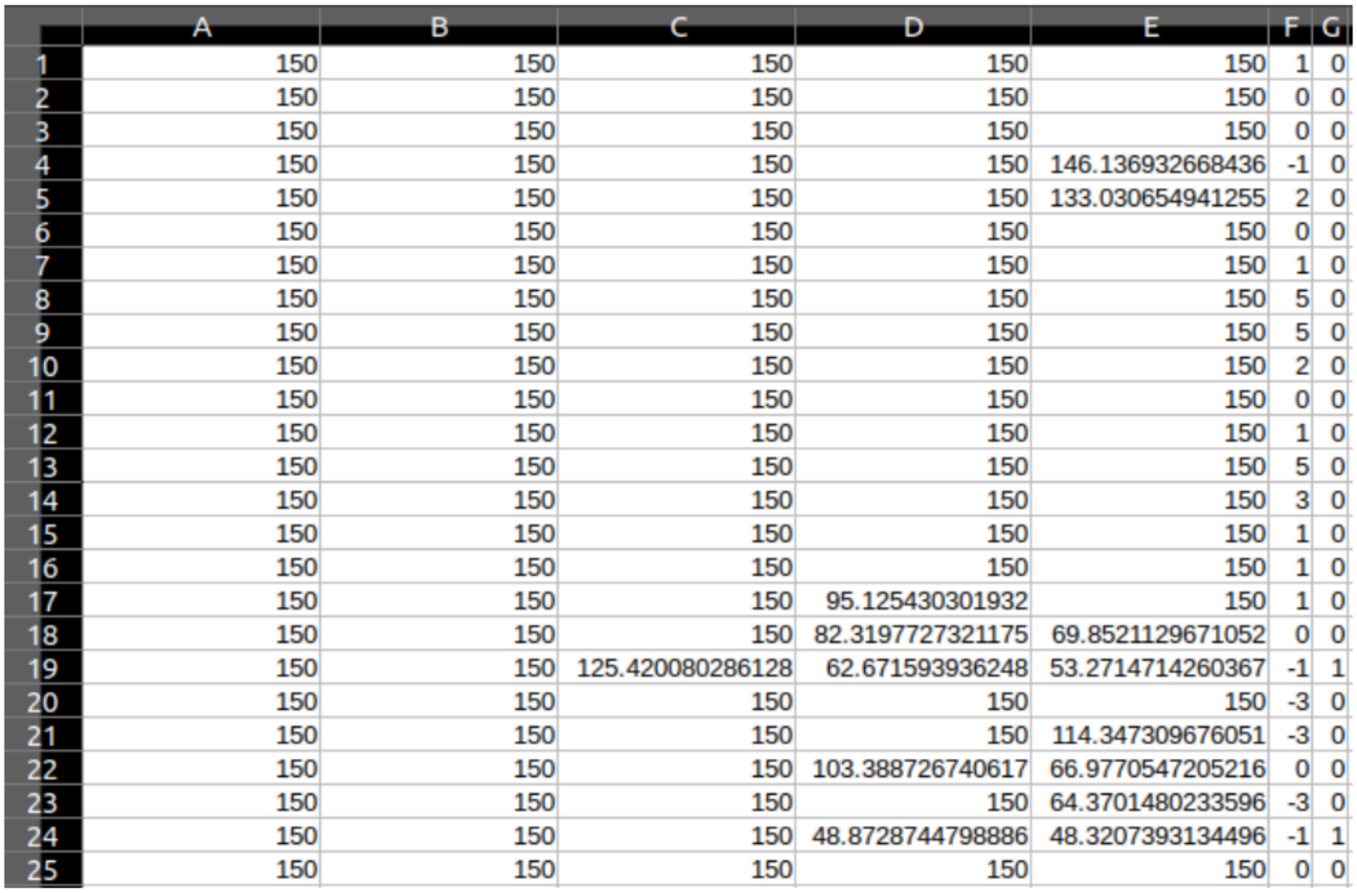


Image 2: Sample Learner Submission.CSV

Image 2: Sample Learner Submission.CSV shows what your submission.csv file should look like with seven (7) untitled columns. You may want to enlarge the image to see it clearly.



	A	B	C	D	E	F	G
1	150	150	150	150	150	1	0
2	150	150	150	150	150	0	0
3	150	150	150	150	150	0	0
4	150	150	150	150	146.136932668436	-1	0
5	150	150	150	150	133.030654941255	2	0
6	150	150	150	150	150	0	0
7	150	150	150	150	150	1	0
8	150	150	150	150	150	5	0
9	150	150	150	150	150	5	0
10	150	150	150	150	150	2	0
11	150	150	150	150	150	0	0
12	150	150	150	150	150	1	0
13	150	150	150	150	150	5	0
14	150	150	150	150	150	3	0
15	150	150	150	150	150	1	0
16	150	150	150	150	150	1	0
17	150	150	150	95.125430301932	150	1	0
18	150	150	150	82.3197727321175	69.8521129671052	0	0
19	150	150	125.420080286128	62.671593936248	53.2714714260367	-1	1
20	150	150	150	150	150	-3	0
21	150	150	150	150	114.347309676051	-3	0
22	150	150	150	103.388726740617	66.9770547205216	0	0
23	150	150	150	150	64.3701480233596	-3	0
24	150	150	150	48.8728744798886	48.3207393134496	-1	1
25	150	150	150	150	150	0	0

Part 2

Download “CSE 571_Part 2_Neural Network for Collision Prediction_Project Files.zip” from the “**Project Overviews and Resources**” page in the *Welcome and Start Here* module, carefully review the directions, and then start working in the provided starting code:

- Data_Loaders.py
- saved/training_data.csv

Now that you have collected your training data from Part 1, you can package it into an iterable PyTorch DataLoader for ease of use. You may be required to prune your collected data to balance out their distribution. If your dataset is 99% 0s and 1% 1s, a model that outputs only 0 would achieve good loss, but it would not have learned anything useful. Make sure to create both a training and testing DataLoader. Use training_data.csv collected from the previous part. Make sure to use the PyTorch classes mentioned in the comments of Data_Loaders.py.

Part 3

Download “CSE 571_Part 3_Neural Network for Collision Prediction_Project Files.zip” from the “**Project Overviews and Resources**” page in the *Welcome and Start Here* module, carefully review the directions, and then start working in the provided starting code:

- Data_Loaders.py
- Networks.py
- saved/*

For Part 3, you will be designing your own custom neural network using PyTorch’s torch.nn class. You will need to initialize a custom architecture, define a forward pass through the network, and build a method for evaluating the fit of a given model.

Part 4

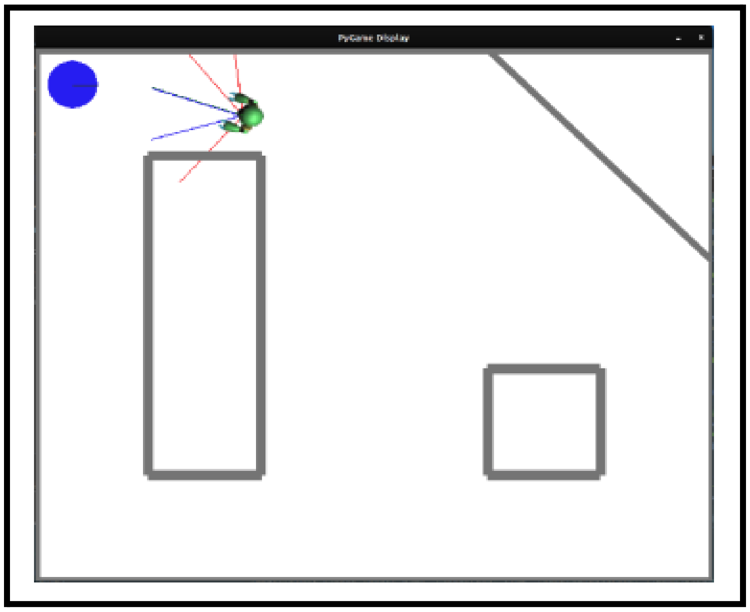
Download “CSE 571_Part 4_Neural Network for Collision Prediction_Project Files.zip” from the “**Project Overviews and Resources**” page in the *Welcome and Start Here* module, carefully review the directions, and then start working in the provided starting code:

- Data_Loaders.py
- Networks.py
- train_model.py
- saved/*

In Part 4, you must train a model using your custom network architecture, which accurately predicts collisions given sensor and action information. Your grade will depend on the accuracy of the model. You may need to try many different strategies and architectures to achieve a well fit model. Keep track of your training and testing loss throughout the epochs, and generate a plot with these lines at the end. To see an application demo of the learning your robot has done, run goal_seeking.py, which will have the robot seek out goals while only taking possible actions it deems to be safe. Please review “Image 1.1: Part 4 PyGame Display.”

Image 1.1: Part 4 PyGame Display

Image 1.1: Part 4 PyGame Display shows the plot of the training and testing loss lines. You may want to enlarge the image to see it clearly.



Submission Directions for Deliverables

You are given an unlimited number of attempts to submit your best work. The number of attempts is given to anticipate any submission errors you may have in regards to properly submitting your best work within the deadline (e.g., accidentally submitting the wrong paper). It is not meant for you to receive multiple rounds of feedback and then one (1) final submission. Only your most recent submission will be assessed.

You must submit the deliverable for each part of the Neural Network for Collision Prediction Project separately through Gradescope. Carefully review submission directions outlined in this overview document in order to correctly earn credit for your work. Learners may not email or use other means to submit any assignment or project for review, including feedback, and grading.

Submitting to Gradescope

Your submission will be reviewed by the course team and then, after the due date has passed, your score will be populated from Gradescope into your Canvas grade.

1. Go to the designated part's submission space in Canvas, "**Submission: Part # - Neural Network for Collision Prediction**".
2. Click the "**Load Submission…in new window**" button.
3. Once in Gradescope, select the part's designated title, "**Submission: Part # - Neural Network for Collision Prediction Project**", and a pop-up window will appear.
4. In the pop-up,
    a. Submit the deliverable(s) for the applicable part of the project.
    b. Click "**Upload**" to submit your work for grading.
5. You will know you have completed the assignment when feedback appears for each test case with a score.
6. If needed: to resubmit the assignment in Gradescope:
    a. Click the "**Resubmit**" button on the bottom right corner of the page and repeat the process from Step 3.

Part 1 Deliverable

Part 1 - Neural Network for Collision Prediction Project includes one (1) deliverable:

- **collect_data.py:** Your collect_data.py script should generate a file named submission.csv that contains 100 data samples in seven (7) untitled columns. For grading purposes, submit your **submission.csv** containing 100 data samples.

Part 2 Deliverable

Part 2 - Neural Network for Collision Prediction Project includes two (2) deliverables:

- **Data_Loaders.py:** This file must be error-free, featuring a Data_Loaders class that initializes with a specified batch size. It ensures that data_loaders.train_loader successfully loads and contains the required data for training purposes.
- **saved/training_data.csv:** This file can be the original one you downloaded from part 1, with 11000 data samples. Keep the file name as "saved/training_data.csv".

Part 3 Deliverable

Part 3 - Neural Network for Collision Prediction Project includes one (1) deliverable:

- **submission.zip:** Your ZIP file should contain the **Data_Loaders.py** and **Networks.py** files.

Part 4 Deliverable

Part 4 - Neural Network for Collision Prediction Project includes one (1) deliverable.

- **submission.zip:** Your ZIP file should contain the **Networks.py** file and **saved/**, which has **saved_model.pkl** and **scaler.pkl** files.

Evaluation

Your submission will be auto-graded. Please review the Evaluation Criteria for how your source code will be assessed. Submissions will be evaluated based on the criteria and will receive a total score. Submissions missing any part of the project will be graded based on what was submitted against the evaluation criteria. Missing parts submitted after the deadline will **not** be graded.

Review the course syllabus for details regarding late penalties.

Evaluation Criteria

Part 1

To receive full points in Part 1, ensure that your .csv file meets the following validation criteria:

- Each row should have exactly 7 values (5 sensor readings, 1 action, 1 collision status).
- Ensure that there are exactly 100 rows of data in the submission file.
- Ensure that the submission data does not contain any `NaN` values.

Part 2

To receive full points in Part 2, ensure that your solution meets the following validation criteria:

- Include **Data_Loaders.py** in the submission.
- Ensure it imports without errors.
- Your **Data_Loaders** class must be correctly defined and instantiated with a batch size.
- Ensure that **data_loaders.train_loader** contains data.
- Each sample in test_loader must have input and label.
- **saved/training_data.csv** is included.

Part 3

To receive full points in Part 3, ensure that your solution zip file meets the following validation criteria:

- Ensure that **Data_Loaders.py** and **Networks.py** should be included in the submission.
- Ensure Action_Conditioned_FF is defined in **Networks.py**.
- Ensure that **Data_Loaders** is correctly defined and can be instantiated.
- Ensure that Action_Conditioned_FF correctly inherits from torch.nn.Module.
- Ensure model.evaluate() can calculate a valid loss on the test loader.

Part 4

To receive full points in Part 4, ensure that your solution zip file meets the following validation criteria:

- Ensure that the **Networks.py** file and **saved/** folder, which has **saved_model.pkl** and **scaler.pkl** file should be included in the submission.
- Ensure Action_Conditioned_FF is defined in **Networks.py**.
- Ensure Action_Conditioned_FF inherits from torch.nn.Module.
- Ensure that your solution files are free from errors.

Your robot will be graded based on the number of collisions and false positives.

- 3% of the maximum score will be deducted for every missed collision above five (5).
- 1% of the maximum score will be deducted for every false positive above ten (10).