Project 3 : Predicting Steps to Locate the Space Rat

Shrey Patel (sp2675)

Abhishek Jani (aj1121)

Mustafa Adil (ma2398)

Table of Contents

1 Introduction

In this project, we simulate a scenario where a bot attempts to catch a space rat hidden within a ship. The bot does not have direct knowledge of the rat's location and must rely on probabilistic sensing to infer its position. Through a series of simulations, we collect data on the ship's layout, the bot's belief about the rat's location, and the steps remaining to catch the rat. We, then use this data to train a convolutional neural network (CNN) to predict the number of steps remaining until the rat is caught.

2 Data Representation

2.1 Input Data

The input of this model consists of two primary components, both represented as two-dimensional grids:

1. **Ship Layout Grid:** A 30×30 grid where each cell indicates whether it is an open space (denoted by 0) or a wall (denoted by 1). This grid layout remains the same across all simulations, giving a fixed environment for the bot and the rat.
2. **Knowledge Probability Grid:** A 30×30 grid representing the bot's current belief about the rat's location. Each cell contains a probability value indicating the likelihood of the rat being present in that cell. Initially, this grid is uniformly distributed but is updated iteratively based on the bot's inputs pings.

We combined these two grids into a two-channel input tensor of shape $(2, 30, 30)$, where the first channel corresponds to the ship layout and the second channel corresponds to the bot's knowledge grid.

2.2 Output Data

Our model outputs a single real-valued number representing the estimated number of steps remaining for the bot to catch the rat. A lower value indicates that the rat is expected to be caught soon, and a higher value suggests that the rat is likely further away or that there is greater uncertainty in the bot's localization.

3 Model Architecture and Design Choices

We have employed a Convolutional Neural Network (CNN), which is designed to leverage spatial hierarchies in the input data. Below is a detailed breakdown of the architecture and the rationale behind each design choice:

3.1 Convolutional Layers

The CNN comprises four convolutional layers, each followed by a ReLU activation function and a max pooling layer:

- **First Convolutional Layer:** It takes the two-channel input and applies 32 filters of size 3×3. The padding ensures that the spatial dimensions are preserved. The ReLU activation introduces non-linearity, allowing the network to learn complex patterns.
- **Second Convolutional Layer:** It increases the depth to 64 filters, maintaining the 3×3 kernel size and padding. This layer captures more intricate features from the input.
- **Third Convolutional Layer:** It expands to 128 filters, continuing with the same kernel size and padding. This deeper layer allows the network to learn higher-level

abstractions.

- **Fourth Convolutional Layer:** It further increases the number of filters to 256, ensuring that the network can capture highly complex features necessary for accurate prediction.

Each convolutional layer is followed by a 2×2 max pooling layer, which reduces the spatial dimensions by half, effectively downsampling the feature maps and reducing computational complexity.

3.2 Fully Connected Layers

After the convolutional layers, the feature maps are flattened and passed through a series of fully connected layers:

- **First Fully Connected Layer:** It connects the flattened convolutional output to 512 neurons, followed by a ReLU activation and a dropout layer with a rate of 0.3 to prevent overfitting.
- **Second Fully Connected Layer:** It reduces the dimensionality to 128 neurons, maintaining the ReLU activation and dropout for continued regularization.
- **Third Fully Connected Layer:** It further reduces to 32 neurons with ReLU activation, preparing the data for the final output layer.
- **Output Layer:** A single neuron with no activation function to output the regression value representing the steps remaining.

The progression from higher to lower-dimensional fully connected layers allows the network to use the extracted features into a meaningful prediction.

4 Loss Function

We have employed Mean Squared Error (MSE) loss function to evaluate the model's performance:

$$\text{MSE} = \frac{1}{N}\sum_{i=1}^{N}(y_i - \hat{y}_i)^2$$

MSE is suitable as it penalizes larger errors more significantly than smaller ones, encouraging the model to produce predictions that are as close as possible to the actual values. Minimizing MSE ensures that the predicted steps remaining closely match the true steps remaining in the simulations.

5 Data Collection Methodology

We have done data collection by running simulations to gather relevant data for training the model. The process is as follows:

5.1 Simulation Setup

Each simulation involves the following steps:

1. **Ship Initialization:** A fixed 30×30 ship grid is initialized to create a perfect maze, ensuring that all open cells are reachable without isolated sections.
2. **Rat Placement:** The rat is randomly placed in one of the open cells within the ship.
3. **Bot Localization:** The bot starts at a localized position inferred from distance signatures relative to the rat's position. The localization algorithm iteratively updates the bot's probability grid based on sensed pings.

4. **Sensing and Movement:** At each step, the bot senses if it hears a "ping." Based on this, it updates its probability grid. The bot then moves to an adjacent open cell randomly.

5. **Data Recording:** For up to 300 steps within a simulation (to prevent infinite loops), the following data points are recorded:

    - **Ship Layout Grid:** Captured at each step to maintain context.
    - **Knowledge Probability Grid:** Updated based on the bot's sensing results.
    - **Steps Remaining:** Calculated using Breadth-First Search (BFS) to determine the shortest path from the bot's current position to the rat.

6. **Termination Conditions:** A simulation concludes when either:

    - The bot catches the rat (i.e., their positions coincide).
    - The simulation reaches 300 steps without catching the rat. To be clear we can easily increase the number of max steps, however our goal is to prevent the model from going into infinite loop.

5.2 Dataset Composition

We ran 1000 simulations, out of which:

- Simulations where the bot successfully catches the rat within 300 steps are included in the dataset.
- Simulations that reach 300 steps without catching the rat are excluded to ensure data quality and relevance.

The resulting dataset comprises numerous states, each containing the ship layout, the bot's knowledge grid, and the steps remaining to catch the rat. This data is subsequently split into training (70%), validation (15%), and testing (15%) sets to train and evaluate the model effectively.

6 Results and Analysis

6.1 Training, Validation, and Test Loss Over Epochs

We trained the model over 20 epochs, with training, validation, and test losses recorded at each epoch. The following figures illustrate the loss trends:

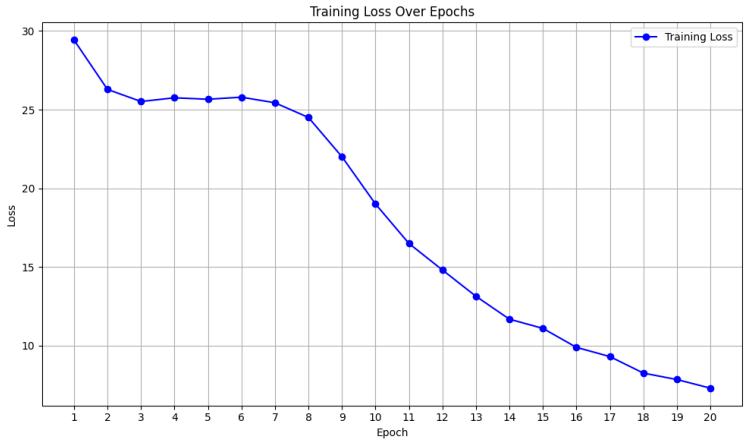

Figure 1: Training Loss Over Epochs

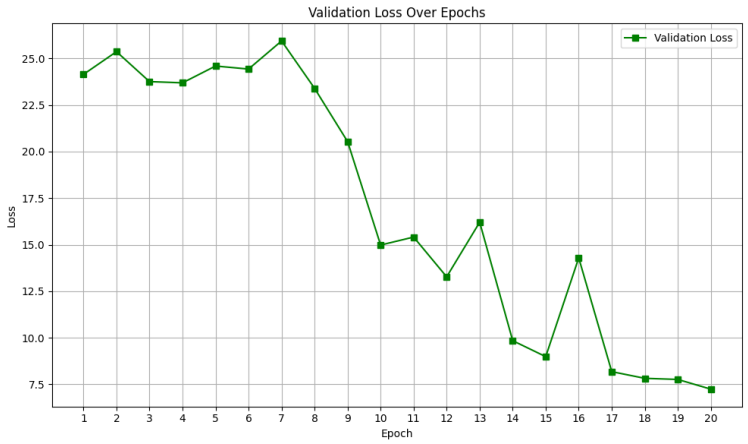

Figure 2: Validation Loss Over Epochs

Figure 3: Test Loss Over Epochs

Analysis:

- **Training Loss:** The training loss decreased steadily over epochs, indicating that the model is learning to minimize errors on the training data.
- **Validation Loss:** Similarly, the validation loss decreases, showing that the model generalizes well to unseen data and is not memorizing the training set.
- **Test Loss:** The test loss also follows a decreasing trend, showing model's ability to perform well on completely unseen data.

Also these losses do not show significant divergence, implying that the model is neither overfitting nor underfitting, achieving a balance between bias and variance.

6.2 Loss as a Function of Steps Remaining

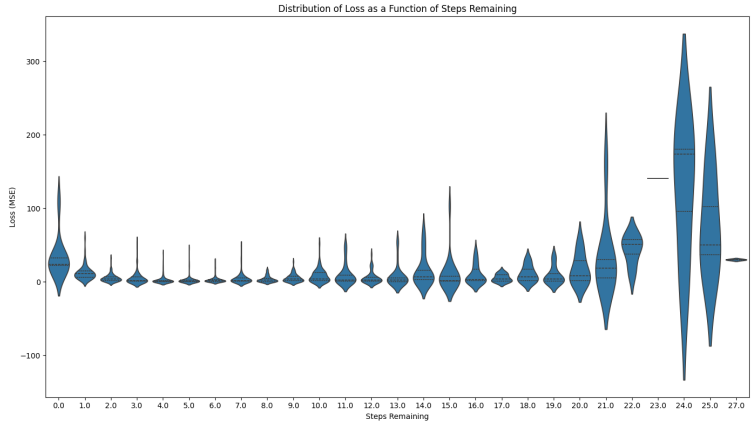

Figure 4: Loss as a Function of Steps Remaining

Intuition and Observation:

- **Early Steps (High Steps Remaining):** The loss is higher, reflecting greater uncertainty and variability in the bot's localization and the potential paths to catch the rat.
- **Later Steps (Low Steps Remaining):** The loss decreases, indicating that the model becomes more confident and accurate as the bot gets closer to the rat.

The initial steps involve more uncertainty, and precision increases as the bot comes near in on the rat's location.

6.3 Distribution of Loss Values

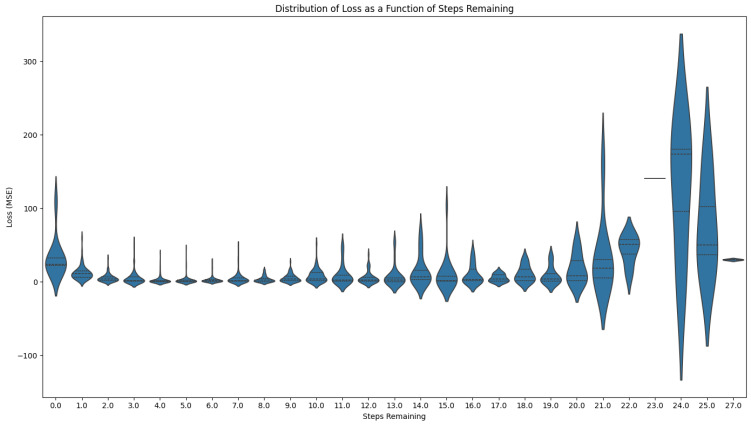

Figure 5: Distribution of Loss as a Function of Steps Remaining (Violin Plot)

Interpretation:

- **Broader Distribution at Higher Steps Remaining:** Indicates high variability and uncertainty in predictions when the bot is far from the rat.
- **Narrower Distribution at Lower Steps Remaining:** Suggests that predictions are more consistent and accurate as the bot approaches the rat.

It emphasizes that the model's performance improves as the simulation progresses towards the rat's location.

6.4 Output Data

The following table presents a snapshot of our simulations results along with the steps remaining for each:

Simulation ID	Starting X	Starting Y	Steps Remaining
101	9	21	12.0
202	26	26	36.0
303	27	13	0.0
404	27	25	45.0
505	2	15	24.0

Table 1: Output of Simulations and Steps Remaining

Observation:

- Simulations with higher steps remaining (e.g., Simulation 404 with 45.0 steps) represent scenarios where the rat is further away or the bot's localization is less precise.
- Simulations with fewer steps remaining (e.g., Simulation 101 with 12.0 steps) indicate that the bot is nearing the rat, aligning with the lower loss values observed in such cases.

7 Conclusion

We successfully demonstrated the application of a Convolutional Neural Network (CNN) in predicting the number of steps remaining for a bot to catch a space rat in a fixed-grid ship environment. Results show that the model learns to reduce training, validation, and test losses over time, providing evidence that the model is learning. The relationship between loss and time remaining in the simulation matches our expectations: higher loss at the start of the simulation, when uncertainty is greater, and lower loss as the bot approaches the rat.