CS - 520

Introduction to Artificial Intelligence

Project 3

Group:

1. **Reo Correia** (rc1460)
2. **Manan Davawala** (md1840)
3. **Vijayendra Sai Chennareddy** (vc545)

Setup:

The grid size of 50 X 50 has been used for the ship layout.
Alien detection sensor: k = 15
Crew detection sensor: ɑ (alpha) = 0.055

Model-1:

Problem Statement:

Given a present state of the simulation, fit a model to predict the next move that the bot takes.
Train the model based on the data generated by repeatedly running Bot-1.

Input space:

The data available to us as input would be the ship layout, probability map of the alien, probability map of the crew, position of the bot, crew detection beep, alien detection beep and possible legal moves. While training, we leave the ship layout out as we have a fixed ship and are also providing the legal moves and bot position. This contains the same essence as the actual layout.

We have taken the above. First we take the two 50X50 probability maps and flatten them. Then we perform some data preprocessing steps on those two 2500 length arrays (Data Preprocessing steps have been explained below). After the preprocessing, the input data has 49 columns. The data has around 60,000 rows.

Data Preprocessing:

First, we use MinMax scaler to scale the data into an appropriate range of 0 to 1.

$$X_{\text{scaled}} = \frac{X - X_{\text{min}}}{X_{\text{max}} - X_{\text{min}}}$$

MinMax Scaling

Next, we employ an auto-encoder for the purpose of dimensionality reduction. We reduce both the crew and alien probability maps to a latent representation of size 25.

After that, we remove any NaN values and apply z-score transformation to bring the mean of the data to 0 and have unit standard deviation.

$$x_{scaled} = \frac{x - mean}{sd}$$

Z-score normalization

Output space:

The output of the model is a one-hot encoded vector of 4 possible classes (left/right/up/down) i.e. (0,1,2,3) respectively.

Model Space:

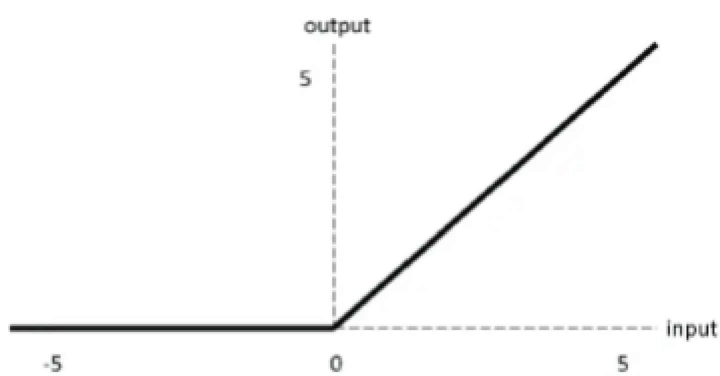
We are using a vanilla neural network for predicting the next move to be taken by the bot.

Features of the neural network:

Number of hidden layers: 5
Dimensions of the hidden layer: 64, 32, 16, 8, 4
I/P layer: Same as the data input size
O/P layer size: 4

Activation Functions:
For all the layers except the o/p layer: ReLU

ReLU (Rectified Linear Activation Function): ReLU is a linear function that retains the input as the output if it is positive, else it will output zero.



ReLU Activation Function

For the o/p layer: Masked Softmax
Masked softmax diverges from the normal softmax as in that it only gives weight to valid predictions. In our case, the model predicting illegal/invalid moves to blocked cells unnecessarily affects the model's performance. We do this by maintaining a list of legal moves for each data point. Later, we multiply it to the predicted values. Valid moves are multiplied by 1, keeping them the same. Whereas, the invalid moves are multiplied by zero, resulting in nullifying their effect. This helps the model make better decisions as it only predicts valid moves.

Loss Functions:

Categorical cross-entropy is being used as the loss function.

$$CE = -\sum_{i=1}^{i=N} y\_true_i \cdot log(y\_pred_i)$$

$$CE = -\sum_{i=1}^{i=N} y_i \cdot log(\widehat{y_i})$$

$$\implies CE = -[y_1 \cdot log(\widehat{y_1}) + y_2 \cdot log(\widehat{y_2}) + y_3 \cdot log(\widehat{y_3})]$$

Categorical Cross-entropy loss function

Categorical Cross-Entropy is a loss function which is used for multi-class classification problems. It measures the difference between the predicted and actual output when the output is categorical. Using this coupled with masked softmax activation function, we get the probabilities of the bot taking the valid moves.

After that, the prediction is made as the output class with the highest probability of occurring.

Training Algorithm: Gradient Descent

Gradient descent function tries to converge to a local minima of any differential function. In any ML model, we try to minimize our cost function.

$$\theta_j = \theta_j - \alpha \frac{\partial}{\partial \theta_j} J(\theta)$$

Gradient descent updation formula

In our model, the derivative of the ReLU function is applied to compute the gradient of the loss with respect to the input of the ReLU-activated neurons.
Weights and biases are both updated in this manner after the backward pass.

<u>Learning Rate:</u> 0.01

Training Results:

```
Epoch 1/50, Training Loss: 2.0785, Training Accuracy: 0.3556, Validation Loss: 1.4873, Validation Accuracy: 0.349
Epoch 2/50, Training Loss: 1.4893, Training Accuracy: 0.3704, Validation Loss: 1.4835, Validation Accuracy: 0.3785
Epoch 3/50, Training Loss: 1.4505, Training Accuracy: 0.4026, Validation Loss: 1.3789, Validation Accuracy: 0.442
Epoch 4/50, Training Loss: 1.3754, Training Accuracy: 0.4546, Validation Loss: 1.3158, Validation Accuracy: 0.4415
Epoch 5/50, Training Loss: 1.3413, Training Accuracy: 0.4677, Validation Loss: 1.1878, Validation Accuracy: 0.4835
Epoch 6/50, Training Loss: 1.3201, Training Accuracy: 0.4809, Validation Loss: 1.1752, Validation Accuracy: 0.478
Epoch 7/50, Training Loss: 1.1999, Training Accuracy: 0.4832, Validation Loss: 1.1478, Validation Accuracy: 0.4725
Epoch 8/50, Training Loss: 1.1835, Training Accuracy: 0.4912, Validation Loss: 1.121, Validation Accuracy: 0.4865
Epoch 9/50, Training Loss: 1.1648, Training Accuracy: 0.4916, Validation Loss: 1.109, Validation Accuracy: 0.496
Epoch 10/50, Training Loss: 1.1448, Training Accuracy: 0.4976, Validation Loss: 1.0645, Validation Accuracy: 0.5085
Epoch 11/50, Training Loss: 1.1295, Training Accuracy: 0.4968, Validation Loss: 1.0748, Validation Accuracy: 0.468
Epoch 12/50, Training Loss: 1.1166, Training Accuracy: 0.4982, Validation Loss: 1.0651, Validation Accuracy: 0.478
Epoch 13/50, Training Loss: 1.1108, Training Accuracy: 0.498, Validation Loss: 1.0384, Validation Accuracy: 0.488
Epoch 14/50, Training Loss: 1.0973, Training Accuracy: 0.4986, Validation Loss: 1.041, Validation Accuracy: 0.5095
Epoch 15/50, Training Loss: 1.0953, Training Accuracy: 0.5054, Validation Loss: 1.0294, Validation Accuracy: 0.504
Epoch 16/50, Training Loss: 1.087, Training Accuracy: 0.5021, Validation Loss: 1.0321, Validation Accuracy: 0.4855
Epoch 17/50, Training Loss: 1.0827, Training Accuracy: 0.4974, Validation Loss: 1.0265, Validation Accuracy: 0.4775
Epoch 18/50, Training Loss: 1.0786, Training Accuracy: 0.4964, Validation Loss: 1.0315, Validation Accuracy: 0.485
Epoch 19/50, Training Loss: 1.0791, Training Accuracy: 0.499, Validation Loss: 1.0281, Validation Accuracy: 0.507
Epoch 20/50, Training Loss: 1.0752, Training Accuracy: 0.4972, Validation Loss: 1.03, Validation Accuracy: 0.486
Epoch 21/50, Training Loss: 1.0703, Training Accuracy: 0.5027, Validation Loss: 1.031, Validation Accuracy: 0.478
```

Epoch 21/50, Training Loss: 1.0703, Training Accuracy: 0.5027, Validation Loss: 1.031, Validation Accuracy: 0.478

Epoch 22/50, Training Loss: 1.0689, Training Accuracy: 0.503, Validation Loss: 1.0216, Validation Accuracy: 0.478

Epoch 23/50, Training Loss: 1.0664, Training Accuracy: 0.5039, Validation Loss: 1.0158, Validation Accuracy: 0.5535

Epoch 24/50, Training Loss: 1.0652, Training Accuracy: 0.5042, Validation Loss: 1.0192, Validation Accuracy: 0.4995

Epoch 25/50, Training Loss: 1.0677, Training Accuracy: 0.5047, Validation Loss: 1.03, Validation Accuracy: 0.575

Epoch 26/50, Training Loss: 1.0605, Training Accuracy: 0.5016, Validation Loss: 1.0607, Validation Accuracy: 0.479

Epoch 27/50, Training Loss: 1.063, Training Accuracy: 0.5057, Validation Loss: 1.0415, Validation Accuracy: 0.4875

Epoch 28/50, Training Loss: 1.061, Training Accuracy: 0.5002, Validation Loss: 1.0263, Validation Accuracy: 0.4995

Epoch 29/50, Training Loss: 1.0594, Training Accuracy: 0.5091, Validation Loss: 1.0125, Validation Accuracy: 0.553

Epoch 30/50, Training Loss: 1.0592, Training Accuracy: 0.5029, Validation Loss: 1.0302, Validation Accuracy: 0.472

Epoch 31/50, Training Loss: 1.0629, Training Accuracy: 0.5042, Validation Loss: 1.0187, Validation Accuracy: 0.503

Epoch 32/50, Training Loss: 1.06, Training Accuracy: 0.501, Validation Loss: 1.0403, Validation Accuracy: 0.465

Epoch 33/50, Training Loss: 1.0553, Training Accuracy: 0.5009, Validation Loss: 1.0365, Validation Accuracy: 0.469

Epoch 34/50, Training Loss: 1.0571, Training Accuracy: 0.5015, Validation Loss: 1.0515, Validation Accuracy: 0.462

Epoch 35/50, Training Loss: 1.0613, Training Accuracy: 0.5026, Validation Loss: 1.0433, Validation Accuracy: 0.463

Epoch 36/50, Training Loss: 1.0564, Training Accuracy: 0.5056, Validation Loss: 1.0317, Validation Accuracy: 0.466

Epoch 37/50, Training Loss: 1.0539, Training Accuracy: 0.5053, Validation Loss: 1.016, Validation Accuracy: 0.5075

Epoch 38/50, Training Loss: 1.0574, Training Accuracy: 0.5051, Validation Loss: 1.0166, Validation Accuracy: 0.4845

Epoch 39/50, Training Loss: 1.0598, Training Accuracy: 0.5049, Validation Loss: 1.0401, Validation Accuracy: 0.505

Epoch 40/50, Training Loss: 1.0529, Training Accuracy: 0.5018, Validation Loss: 1.02, Validation Accuracy: 0.551

Epoch 40/50, Training Loss: 1.0529, Training Accuracy: 0.5018, Validation Loss: 1.02, Validation Accuracy: 0.551

Epoch 41/50, Training Loss: 1.0558, Training Accuracy: 0.5013, Validation Loss: 1.0204, Validation Accuracy: 0.4975

Epoch 42/50, Training Loss: 1.0566, Training Accuracy: 0.5005, Validation Loss: 1.0228, Validation Accuracy: 0.572

Epoch 43/50, Training Loss: 1.0523, Training Accuracy: 0.5049, Validation Loss: 1.034, Validation Accuracy: 0.484

Epoch 44/50, Training Loss: 1.0517, Training Accuracy: 0.5042, Validation Loss: 1.0393, Validation Accuracy: 0.453

Epoch 45/50, Training Loss: 1.0536, Training Accuracy: 0.509, Validation Loss: 1.0158, Validation Accuracy: 0.572

Epoch 46/50, Training Loss: 1.0543, Training Accuracy: 0.5039, Validation Loss: 1.0101, Validation Accuracy: 0.5775

Epoch 47/50, Training Loss: 1.0548, Training Accuracy: 0.5048, Validation Loss: 1.0213, Validation Accuracy: 0.575

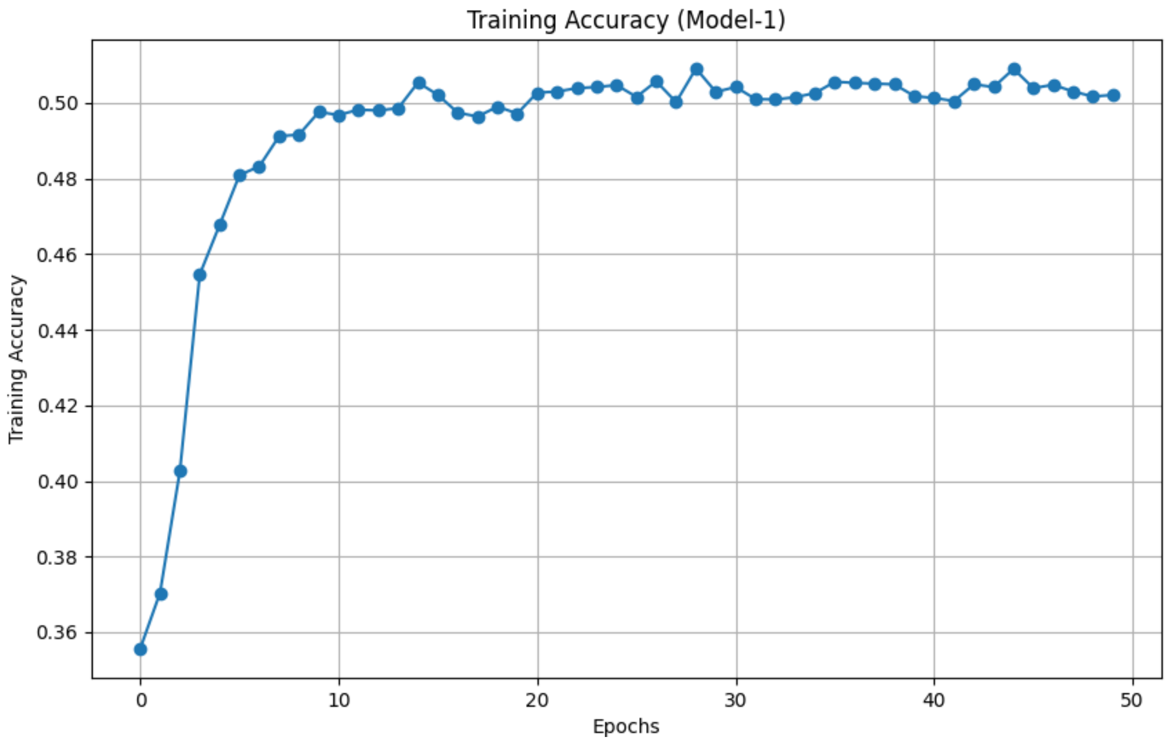
Epoch 48/50, Training Loss: 1.0557, Training Accuracy: 0.503, Validation Loss: 1.0285, Validation Accuracy: 0.4855

Epoch 49/50, Training Loss: 1.0549, Training Accuracy: 0.5017, Validation Loss: 1.0304, Validation Accuracy: 0.4665

Epoch 50/50, Training Loss: 1.0578, Training Accuracy: 0.5021, Validation Loss: 1.0219, Validation Accuracy: 0.5555

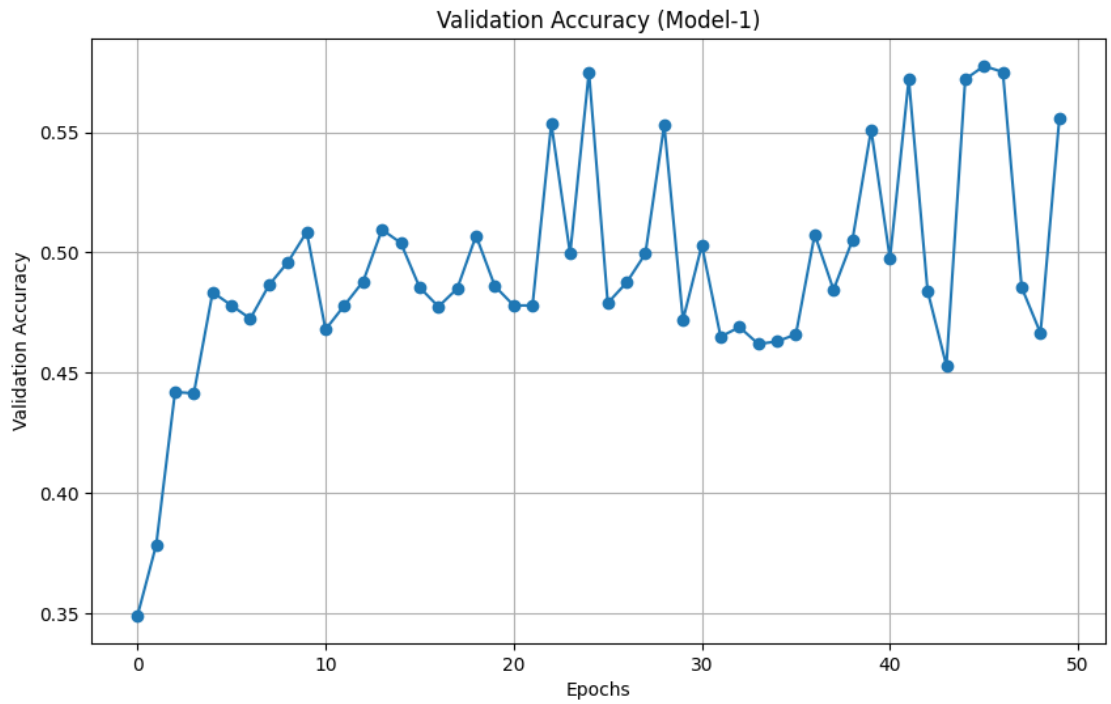
Test Loss: 1.0528, Test Accuracy: 0.5621

Training Accuracy visualization:



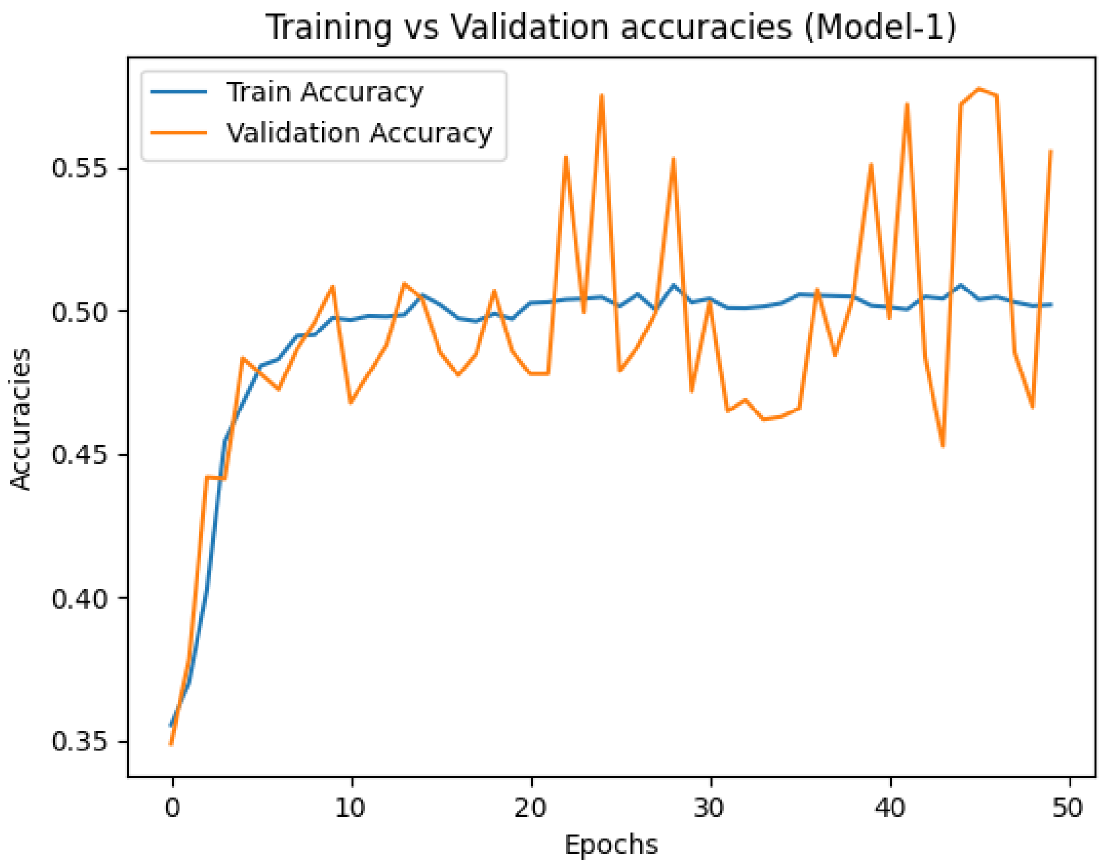
We can see that the model learns most of the details in the first 15 epochs of the training, where it has significant jumps in accuracy over each epoch.
Later on, it can be seen that the model has reached a local maxima as it changes a little bit but in general, it remains in the same small range.

Validation Accuracy visualization:



Similar conclusions can be drawn for the validation accuracy which is on unseen data for the model.

Training vs Validation Accuracy (Proof of model not overfiiting):



The above image is a comparison between the training and validation accuracy. As can be seen, they have a similar pattern of growth. When the training and validation accuracy grow similarly, it is a sign that the model is not overfitting and is actually learning features, not just memorizing them. This can be concluded as the model achieves similar accuracy on the unseen validation data as well.

Performance review:

The model reaches an accuracy of around 56%. This is inferior as compared to Bot-1. It will, 56% of the time, take the same steps as the Bot-1. Hence, a bot following the steps predicted by the model will make the steps as taken by the Bot-1 56% of the time.

Model-2:

Problem statement:

Based on the current state of the bot, predict the probability that the bot saves the crew.

Input space:

In addition to the input given to the model 1, we attach the move that the bot-1 takes at that state. Thus, we have combined input of the alien probability map, crew probability map, bot position and the move.

The data preprocessing step remains the same as that in Model-1

Output space:

A percentage value of the probability of the bot saving the crew based on the state input.

Model space:

After trying out several ML models like: Logistic regression, Decision tree, Random forest, Support vector machine [SVM], the neural network still gave the best results for this task of predicting percentages.

Features of the neural network:

Number of hidden layers: 5
Dimensions of the hidden layer: 64, 32, 16, 8, 4
I/P layer: Same as the data input size
O/P layer size: 1
Activation Functions:
For all the layers except the o/p layer: ReLU
For o/p layer: Sigmoid

Sigmoid has a characteristic of an S-shaped curve. This allows the network to introduce non-linearity into the model. This in turn helps the neural network to learn more complex and convoluted decision boundaries.
In our case, the most helpful property of Sigmoid is that it lies in the range of (0,1). This helps us in our case where we need to predict probability, a number between 0 and 1. By applying a threshold of 0.5 on the output, it gives us the predicted output label as well (saved or not saved).

$$S(x) = \frac{1}{1 + e^{-x}}$$

Sigmoid Activation Function

Loss Function:

Here, the output layer has only 1 neuron. Thus, in a sense it's a binary classification (0/1). Thus, the most appropriate loss function would be binary cross-entropy.

$$L_{BCE} = -\frac{1}{n}\sum_{i=1}^{n}(Y_i \cdot \log \hat{Y}_i + (1 - Y_i) \cdot \log (1 - \hat{Y}_i))$$

Binary Cross-Entropy Loss Function

Training algorithm: Gradient descent

Learning rate: 0.01

Avoiding success being overrepresented in the training:

We know that Bot-1 is extremely successful in saving the crew. To avoid the success being overrepresented while training the model, we employed the method of Random Oversampling. What Random oversampling does is that it randomly selects data points from the minority class and adds them to the training dataset. This duplication of minority class in the training dataset increases the representation of failure in our dataset.

Training Results:

```
Epoch 1/50, Training Loss: 0.283, Training Accuracy: 0.462, Validation Loss: 0.2784, Validation Accuracy: 0.4688
Epoch 2/50, Training Loss: 0.269, Training Accuracy: 0.4685, Validation Loss: 0.2396, Validation Accuracy: 0.4892
Epoch 3/50, Training Loss: 0.2937, Training Accuracy: 0.5532, Validation Loss: 0.2694, Validation Accuracy: 0.8604
Epoch 4/50, Training Loss: 0.2939, Training Accuracy: 0.5484, Validation Loss: 0.4852, Validation Accuracy: 0.7759
Epoch 5/50, Training Loss: 0.2882, Training Accuracy: 0.6577, Validation Loss: 0.2727, Validation Accuracy: 0.8682
Epoch 6/50, Training Loss: 0.2776, Training Accuracy: 0.6656, Validation Loss: 0.275, Validation Accuracy: 0.8676
Epoch 7/50, Training Loss: 0.2782, Training Accuracy: 0.7634, Validation Loss: 0.295, Validation Accuracy: 0.852
Epoch 8/50, Training Loss: 0.2787, Training Accuracy: 0.7662, Validation Loss: 0.3065, Validation Accuracy: 0.8448
Epoch 9/50, Training Loss: 0.3044, Training Accuracy: 0.7478, Validation Loss: 0.2438, Validation Accuracy: 0.876
Epoch 10/50, Training Loss: 0.2687, Training Accuracy: 0.7683, Validation Loss: 0.2675, Validation Accuracy: 0.858
Epoch 11/50, Training Loss: 0.2811, Training Accuracy: 0.8622, Validation Loss: 0.2589, Validation Accuracy: 0.8754
Epoch 12/50, Training Loss: 0.2761, Training Accuracy: 0.8634, Validation Loss: 0.2767, Validation Accuracy: 0.8712
Epoch 13/50, Training Loss: 0.2681, Training Accuracy: 0.8701, Validation Loss: 0.2627, Validation Accuracy: 0.8784
Epoch 14/50, Training Loss: 0.2753, Training Accuracy: 0.8632, Validation Loss: 0.2381, Validation Accuracy: 0.8862
Epoch 15/50, Training Loss: 0.2663, Training Accuracy: 0.8681, Validation Loss: 0.2669, Validation Accuracy: 0.8784
Epoch 16/50, Training Loss: 0.2766, Training Accuracy: 0.865, Validation Loss: 0.2498, Validation Accuracy: 0.8688
Epoch 17/50, Training Loss: 0.2646, Training Accuracy: 0.8707, Validation Loss: 0.2559, Validation Accuracy: 0.8724
Epoch 18/50, Training Loss: 0.2659, Training Accuracy: 0.8698, Validation Loss: 0.2667, Validation Accuracy: 0.8784
Epoch 19/50, Training Loss: 0.2895, Training Accuracy: 0.854, Validation Loss: 0.2559, Validation Accuracy: 0.8754
Epoch 20/50, Training Loss: 0.2931, Training Accuracy: 0.855, Validation Loss: 0.287, Validation Accuracy: 0.8544
Epoch 21/50, Training Loss: 0.2719, Training Accuracy: 0.8709, Validation Loss: 0.2975, Validation Accuracy: 0.8544
```

Epoch 21/50, Training Loss: 0.2719, Training Accuracy: 0.8709, Validation Loss: 0.2975, Validation Accuracy: 0.8544

Epoch 22/50, Training Loss: 0.2761, Training Accuracy: 0.8612, Validation Loss: 0.3181, Validation Accuracy: 0.834

Epoch 23/50, Training Loss: 0.2764, Training Accuracy: 0.8636, Validation Loss: 0.2598, Validation Accuracy: 0.8646

Epoch 24/50, Training Loss: 0.2736, Training Accuracy: 0.8639, Validation Loss: 0.259, Validation Accuracy: 0.8646

Epoch 25/50, Training Loss: 0.2689, Training Accuracy: 0.8662, Validation Loss: 0.2486, Validation Accuracy: 0.888

Epoch 26/50, Training Loss: 0.2744, Training Accuracy: 0.8648, Validation Loss: 0.2441, Validation Accuracy: 0.8766

Epoch 27/50, Training Loss: 0.2697, Training Accuracy: 0.868, Validation Loss: 0.2944, Validation Accuracy: 0.8496

Epoch 28/50, Training Loss: 0.2727, Training Accuracy: 0.864, Validation Loss: 0.2598, Validation Accuracy: 0.8748

Epoch 29/50, Training Loss: 0.2713, Training Accuracy: 0.8647, Validation Loss: 0.2963, Validation Accuracy: 0.849

Epoch 30/50, Training Loss: 0.2841, Training Accuracy: 0.8606, Validation Loss: 0.3146, Validation Accuracy: 0.8436

Epoch 31/50, Training Loss: 0.2666, Training Accuracy: 0.8705, Validation Loss: 0.2576, Validation Accuracy: 0.8754

Epoch 32/50, Training Loss: 0.2715, Training Accuracy: 0.865, Validation Loss: 0.2688, Validation Accuracy: 0.8634

Epoch 33/50, Training Loss: 0.2625, Training Accuracy: 0.8737, Validation Loss: 0.2356, Validation Accuracy: 0.8868

Epoch 34/50, Training Loss: 0.275, Training Accuracy: 0.8638, Validation Loss: 0.2674, Validation Accuracy: 0.8652

Epoch 35/50, Training Loss: 0.2839, Training Accuracy: 0.8614, Validation Loss: 0.277, Validation Accuracy: 0.87

Epoch 36/50, Training Loss: 0.2979, Training Accuracy: 0.8506, Validation Loss: 0.3233, Validation Accuracy: 0.8394

Epoch 37/50, Training Loss: 0.275, Training Accuracy: 0.8645, Validation Loss: 0.2741, Validation Accuracy: 0.8628

Epoch 38/50, Training Loss: 0.2735, Training Accuracy: 0.859, Validation Loss: 0.253, Validation Accuracy: 0.8832

Epoch 39/50, Training Loss: 0.2684, Training Accuracy: 0.8674, Validation Loss: 0.2844, Validation Accuracy: 0.8574

Epoch 40/50, Training Loss: 0.2754, Training Accuracy: 0.8678, Validation Loss: 0.3258, Validation Accuracy: 0.8352

Epoch 41/50, Training Loss: 0.2726, Training Accuracy: 0.8661, Validation Loss: 0.2605, Validation Accuracy: 0.8682

Epoch 40/50, Training Loss: 0.2754, Training Accuracy: 0.8678, Validation Loss: 0.3258, Validation Accuracy: 0.8352

Epoch 41/50, Training Loss: 0.2726, Training Accuracy: 0.8661, Validation Loss: 0.2605, Validation Accuracy: 0.8682

Epoch 42/50, Training Loss: 0.2749, Training Accuracy: 0.8663, Validation Loss: 0.2631, Validation Accuracy: 0.8802

Epoch 43/50, Training Loss: 0.2813, Training Accuracy: 0.8624, Validation Loss: 0.2664, Validation Accuracy: 0.8832

Epoch 44/50, Training Loss: 0.2585, Training Accuracy: 0.8775, Validation Loss: 0.3087, Validation Accuracy: 0.846

Epoch 45/50, Training Loss: 0.2651, Training Accuracy: 0.8728, Validation Loss: 0.2757, Validation Accuracy: 0.8598

Epoch 46/50, Training Loss: 0.259, Training Accuracy: 0.8736, Validation Loss: 0.3021, Validation Accuracy: 0.8448

Epoch 47/50, Training Loss: 0.2701, Training Accuracy: 0.8686, Validation Loss: 0.3019, Validation Accuracy: 0.864

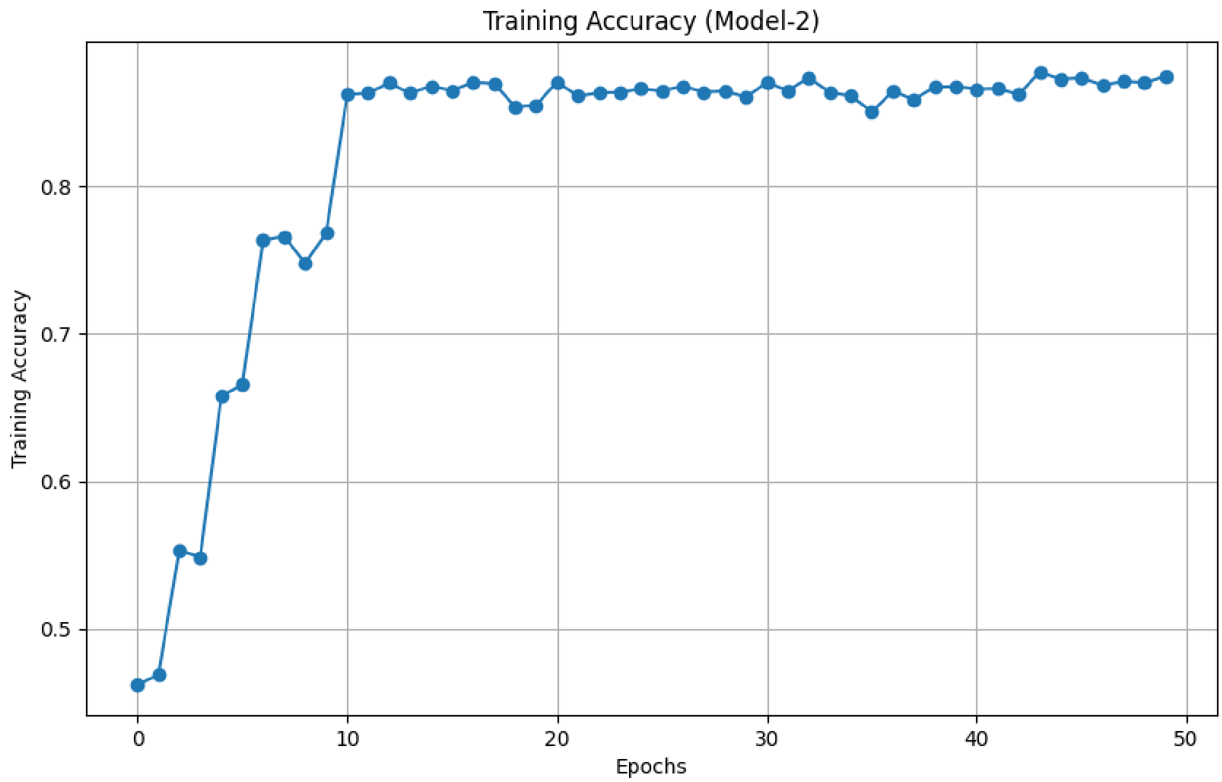
Epoch 48/50, Training Loss: 0.2637, Training Accuracy: 0.8713, Validation Loss: 0.253, Validation Accuracy: 0.876

Epoch 49/50, Training Loss: 0.2643, Training Accuracy: 0.8704, Validation Loss: 0.2464, Validation Accuracy: 0.8898

Epoch 50/50, Training Loss: 0.2589, Training Accuracy: 0.8747, Validation Loss: 0.2535, Validation Accuracy: 0.8939

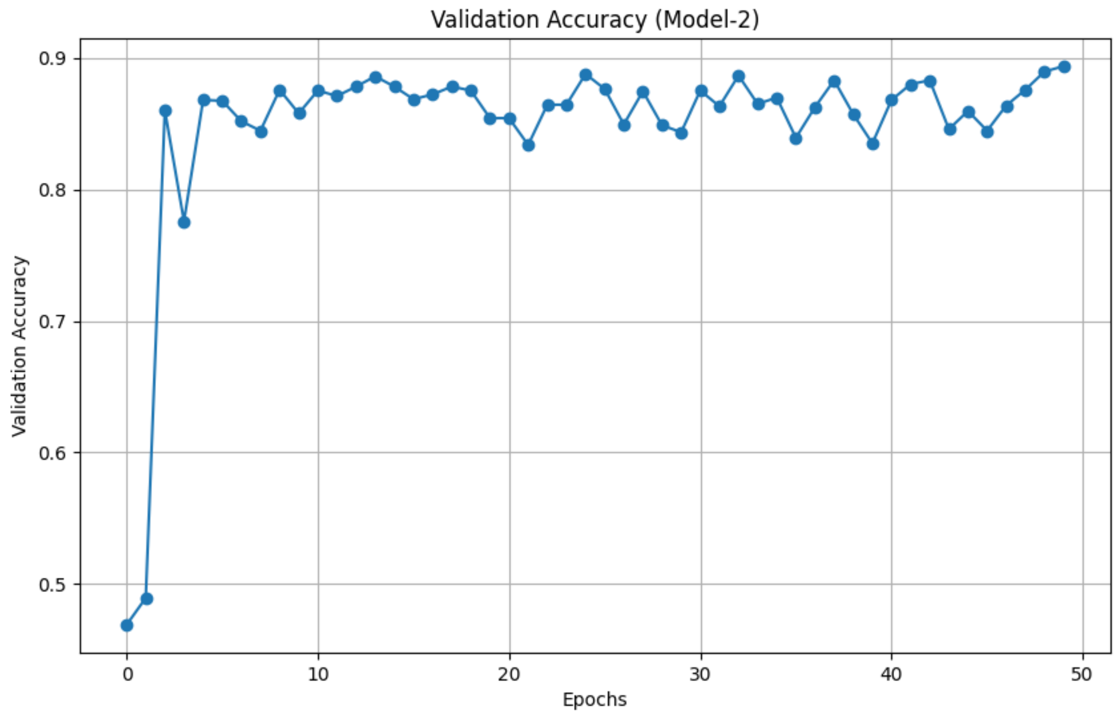
Test Loss: 0.2369, Test Accuracy: 0.8915

Training Accuracy Visualization;



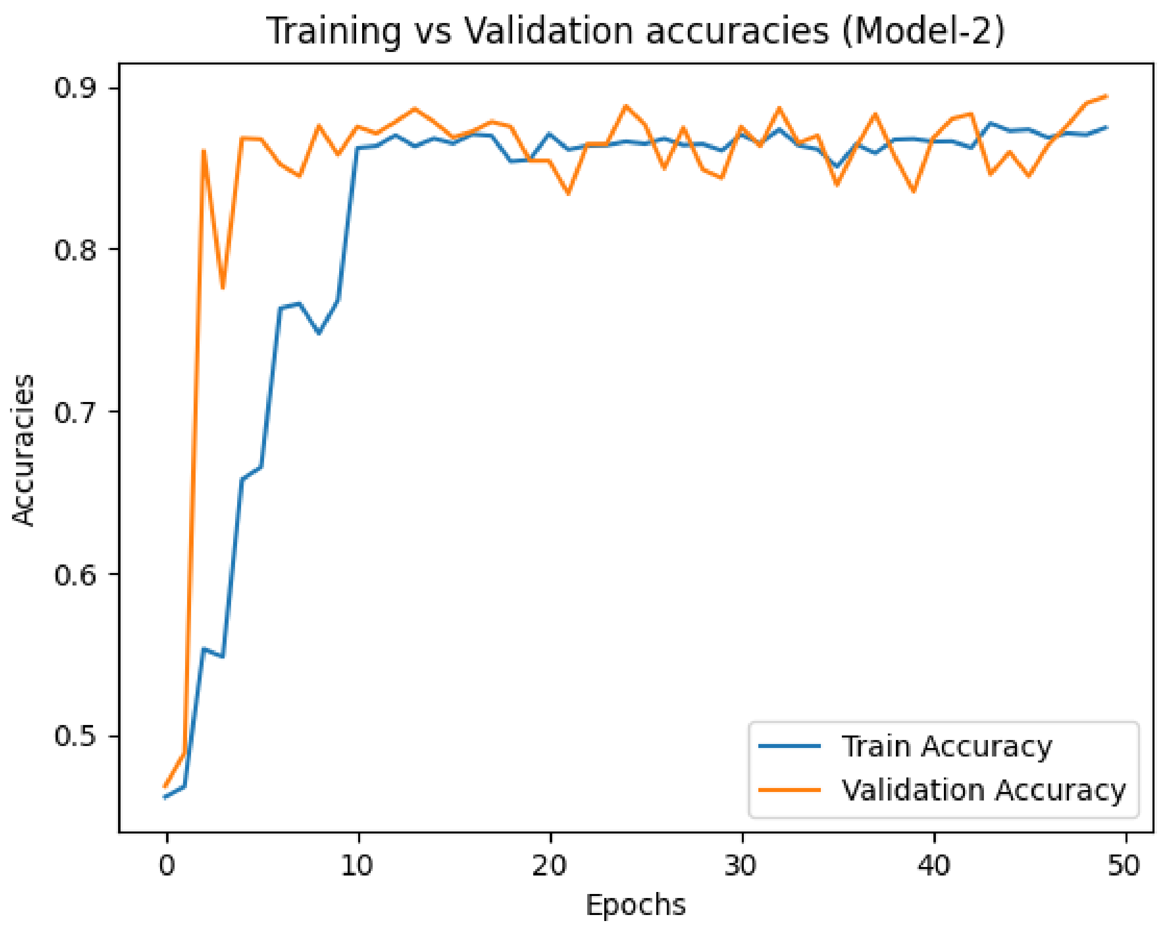
Similar to Model-1, we can see that the model learns most of the details in the first 15 epochs of the training. During this period there is a high jump in the training accuracy. After this, once the local maxima is reached, it plateaus off.

Validation Accuracy Visualization:



Similar behavior is depicted by the validation accuracy as well.

Training vs Validation accuracy Visualization (Proof of model not overfitting):



Again, as the validation accuracy increases with the training accuracy, we can be sure that the model is not overfitting as it performs equally well for unseen data as well.

Model-3

Model specification:

ACTOR Model: LogisticRegression(multi_class='ovr', solver='lbfgs', max_iter=1000, warm_start=True)

The actor model is a Logistic Regression model with the following parameters:

multi_class: This parameter defines the strategy for handling multiple classes in a classification problem. In this case, it is set to 'ovr', which stands for "one-vs-rest." In the one-vs-rest strategy, a binary logistic regression model is fit for each class, treating that class as the positive class and the rest as the negative class.

solver: The solver parameter defines the optimization algorithm used for fitting the logistic regression model. The 'lbfgs' solver proved to be a good optimization algorithm for small to medium-sized datasets. This influences the convergence and performance of the logistic regression model.

max_iter: This parameter specifies the maximum number of iterations for the solver to converge.

warm_start: This parameter enables incremental learning for the model. Due to this, the model retains the existing solution of the previous call to fit() and continues training on the new data. Hence, each iteration adds to the previous learned weights rather than initializing new ones. This is crucial for our case where both, ACTOR and CRITIC, keep on learning incrementally.

CRITIC Model: RandomForestClassifier(warm_start=True)

Random Forest Classifier is a good choice for binary classification because it is an ensemble method that combines multiple decision trees. It generally provides robustness against overfitting. Its nonlinear nature makes it suitable for capturing complex relationships in the data.

Approach:

The Actor model initially mimics the Bot-1. Hence, it takes in input the state of the simulation and predicts the next move.
The Critic takes in the current state as well the move taken by the Actor, then it predicts the performance of the Actor. It does this by predicting the probability of saving the crew, had that step been taken by the bot. Then, it provides the move with the highest probability of saving the crew. This is done by predicting the probability of saving the crew for every move and then selecting the one with the highest probability.
This information is fed to the Actor model as its training data. Following this, the actor model keeps improving.
Also, the more the critic model is trained, the better it gets at learning the performance of the Actor model. This in turn further improves the actor model.

This symbiotic relationship between the actor and the critic gives rise to a ladder-like learning where both the models keep getting better off of each other.

Showing that the ACTOR network is successfully learning the CRITIC-specified actions:

The ACTOR is trained on the data provided by the CRITIC model. Hence, the touchstone to check if the actor network is learning from the critic-specified actions is to make sure that the training accuracy of the actor model keeps on increasing.
This can be confirmed from the accuracies shown below (at the end)

Showing that the ACTOR network is improving based on this process:

The ACTOR after being trained is tested on the actual Bot-1 data. Hence, to make sure that in this entire process the actor gets improved, the testing accuracy of the actor should keep on improving. This can be confirmed from the accuracies shown below (at the end)

Showing that the CRITIC network is successfully learning the performance of the ACTOR network:

As can be seen from the data above, with each epoch the accuracy of the critic model improves. Along with that, the training accuracy of the actor model also improves.

Thus, with improvement of the critic model, the actor model improves too. This proves that the critic is getting better at learning the performance of the actor. Due to this, the actor in turn performs better as well.

Below are the final accuracies of the actor and critic model after each epoch:

```
Epoch:  1
Critic Accuracy:  0.7794
Actor Accuracy (Train):  0.361
Actor Accuracy (Test):  0.3104


Epoch:  2
Critic Accuracy:  0.7951
Actor Accuracy (Train):  0.4436
Actor Accuracy (Test):  0.3221


Epoch:  3
Critic Accuracy:  0.8061
Actor Accuracy (Train):  0.5058
Actor Accuracy (Test):  0.3981


Epoch:  4
Critic Accuracy:  0.8338
Actor Accuracy (Train):  0.5088
Actor Accuracy (Test):  0.4123


Epoch:  5
Critic Accuracy:  0.8352
Actor Accuracy (Train):  0.5129
Actor Accuracy (Test):  0.4162


Epoch:  6
Critic Accuracy:  0.8387
Actor Accuracy (Train):  0.5213
Actor Accuracy (Test):  0.4339
```

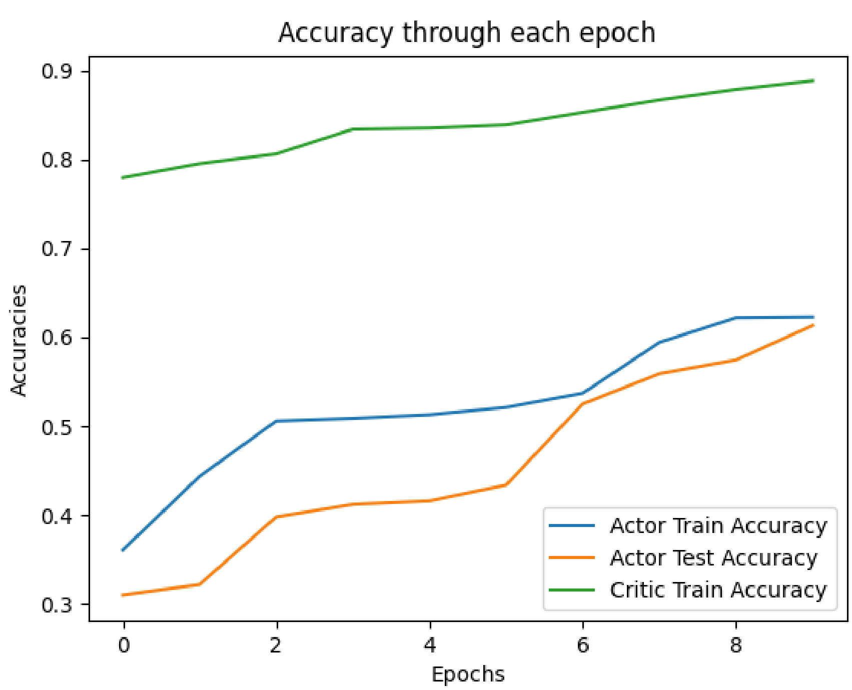
```
Epoch:  7
Critic Accuracy:  0.8527
Actor Accuracy (Train):  0.5371
Actor Accuracy (Test):  0.5251


Epoch:  8
Critic Accuracy:  0.8666
Actor Accuracy (Train):  0.5942
Actor Accuracy (Test):  0.5591


Epoch:  9
Critic Accuracy:  0.8785
Actor Accuracy (Train):  0.622
Actor Accuracy (Test):  0.5745


Epoch:  10
Critic Accuracy:  0.8882
Actor Accuracy (Train):  0.6228
Actor Accuracy (Test):  0.6133
```

Actor and Critic Accuracy Visualization:



Can you beat the performance of Bot 1?

As can be seen from the above statistic, the Actor reaches about 60% performance of Bot-1. Thus, although the actor model has seen considerable improvement from its starting state, it yet cannot outperform Bot-1.

With more computational resources and a higher volume of data, we can aspire to beat the performance of Bot-1, where the Critic model becomes better than the path-finding algorithm of Bot-1.