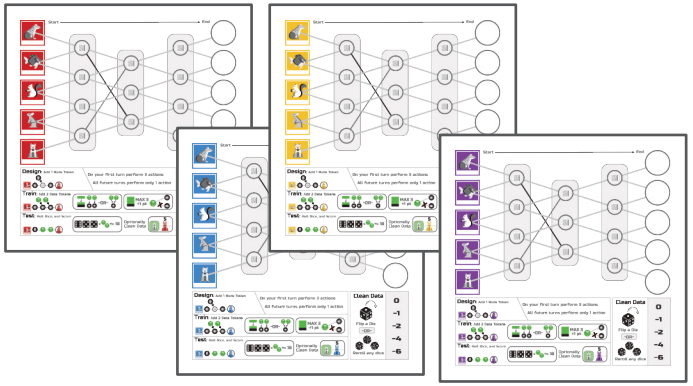
Overview

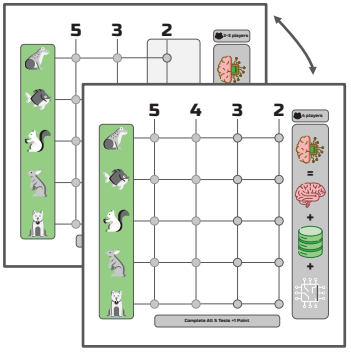
You are a data scientist at FuzzNet Labs a startup creating a service for identifying cute animals in photos. This is becoming a hot market so you will need to compete against the other startups in this space. Whoever gets to market first will have an advantage, but at the end of the day the best model will win.

Components

4 Player Boards



1 Score Board



Animal Figures

1 First Player Marker

4 Data Clean Token

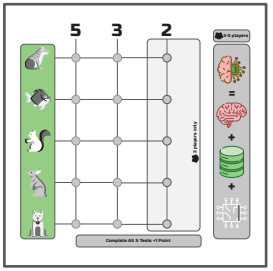
44 Node Tokens

15 Overfitting Tokens

100 Data Tokens

Set Up

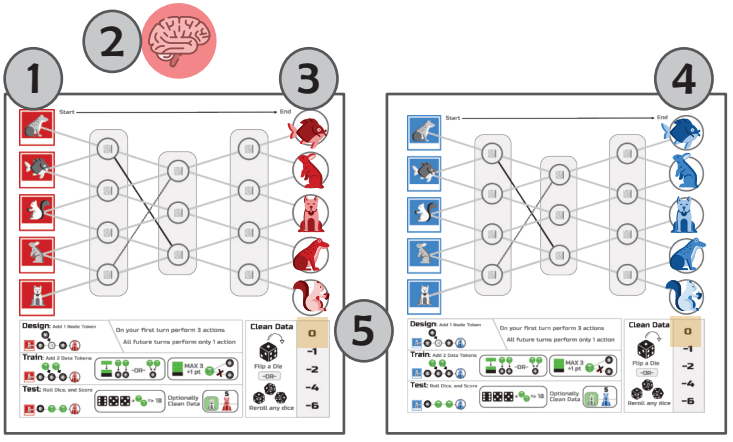
Center of Table Set Up



All Tokens

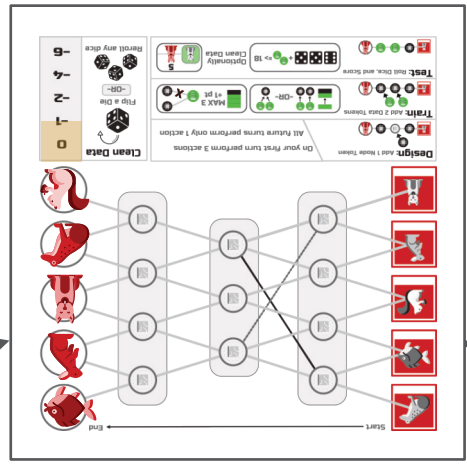
1. Place the Score Board in the middle of the table. Place the side face up that matches the number of players for your game.
2. Place the dice, Node Tokens, Data Tokens and Overfitting Tokens within arms reach of all players.

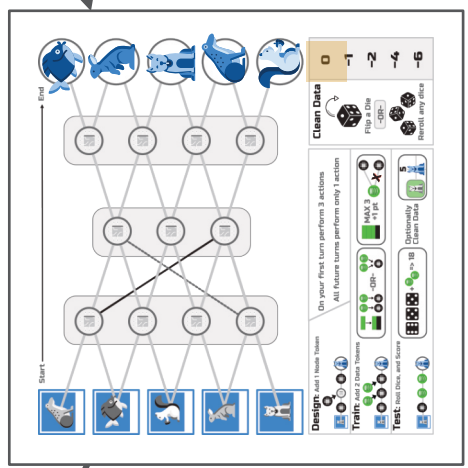
Player Set Up

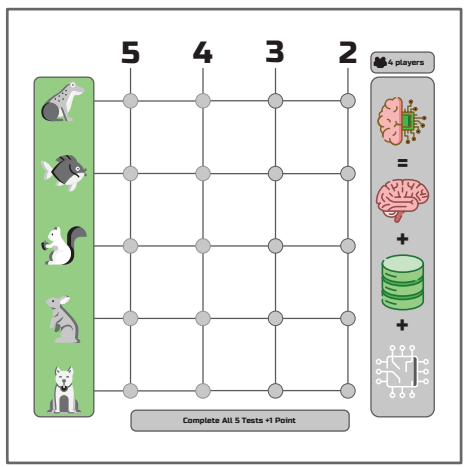


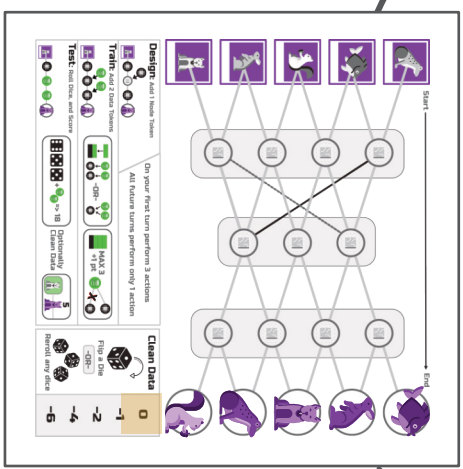
1. Each player receives a player board, and Animal Tokens of a single matching color.
2. Determine who will go first by the method of your choice. Give a player the First Player Marker.
3. The first player places all of their Animal Tokens on their board on the right most column in any order.
4. All players must set up their Animal Tokens on their Player Boards to match the first players' board.
5. Each player places a Data Clean Token at the 0 place on the data clean tracker.

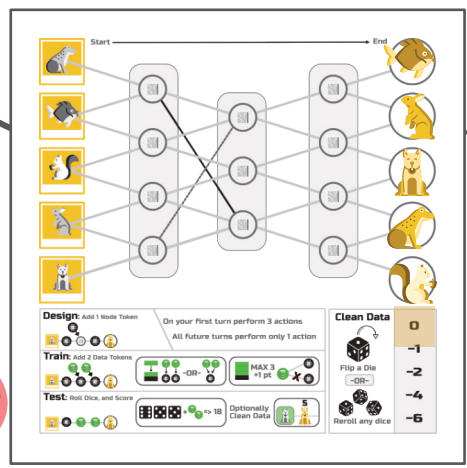
Set Up Example











Have within reach

x44

x100

X x15

Gameplay Overview

Play starts with the first player and continues in clockwise order around the table.

On a players first turn they will take 3 actions, and then all future turns they only take 1 action.

Actions a player can take:

- Design Model - Enables you to perform a test for a specific animal.
- Train Model - Increases the probability of a successful test.
- Test Model - Test a specific animal to score points.

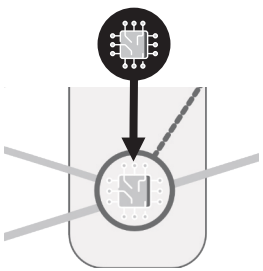
Game end is triggered by either:

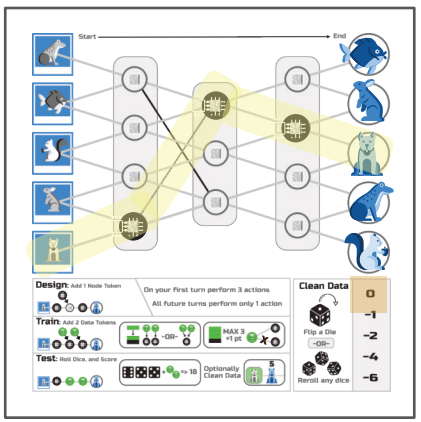
- A player successfully pass tests for all 5 of their Animal Tokens.
- Testing becomes impossible. This occurs if players do too much training and create too many blocked paths through overfitting. Explained on page 3.

After game end is triggered, play continues clockwise until all players have had the same number of turns.

Design Model

Add 1 Node Token on any empty node on your Player Board. This creates a valid test path, explained more on page 4. A Node Token cannot be removed or transferred to another node once it is placed.

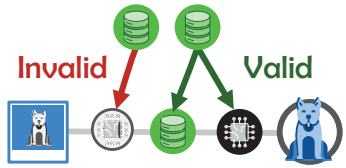




Train Model

Train a model by adding data to it, so you increase your odds of a successful test.

- Add 2 Data Tokens on top of Node Tokens on your Player Board. Data Tokens cannot be removed or transferred after they are placed.



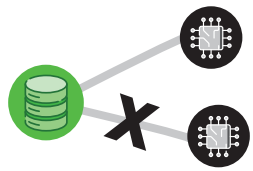
- Data Tokens may be placed on the same node or on 2 different nodes.

For each Data Token on a node will add +1 when calculating a successful test. This increases the probability of a successful test. Explained on page 4.

Overfitting

Max 3 Data Tokens per node. This means you have a highly trained node, but there might be an overfitting problem. Overfitting is when the data in your model is too heavily biased to give accurate responses.

- Place an Overfitting Token on one of the lines/paths from the maxed node to one of the next nodes of your choice. This makes it so this path cannot be used for testing for the rest of the game.

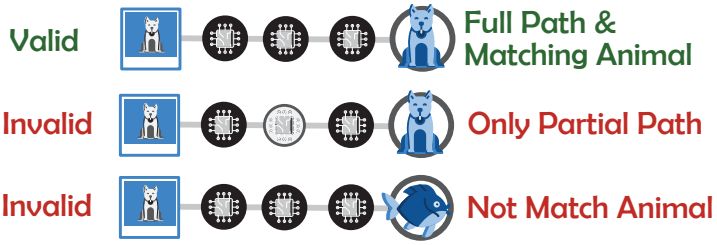


- You get +1 point for each maxed out node for having a highly trained model.

- If a path is no longer needed because you have already successfully tested a specific animal there is no downside of blocking off the path.
- It is possible to block off all paths that can lead to testing a specific animal, so be careful.

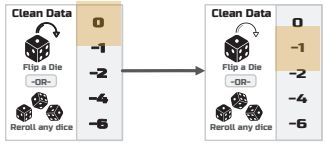
Test Model

1. Identify a path to test, which are 3 nodes with Node Tokens that connect an animal on the left of the player board with the matching Animal Token.



2. Roll the 3 dice.
3. You may use the cleaning data action to modify the value of your dice:
   1. Either reroll any number of dice or flip a single die (1->6, 2 ->5, 3->4).

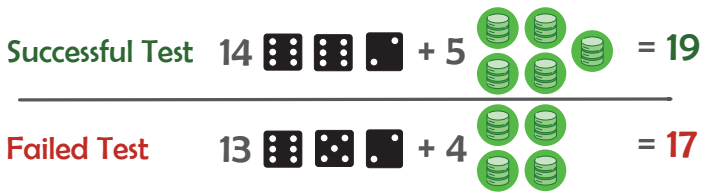
   2. Then slide the Clean Data Token down 1 space on your player board.



Each player can do this action a maximum of 4 times per game. Each time you use the clean data action it creates negative points at the end of the game. You should have used clean data before testing!

4. Calculate test status = dice value + # of Data Tokens on the testing path.

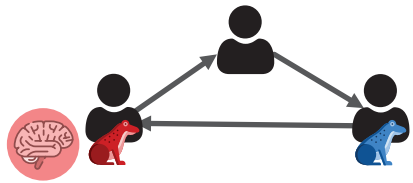
   A test passes with a value of **18 or more.**

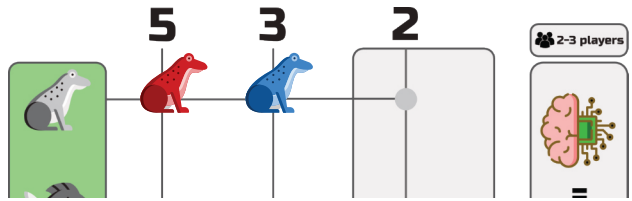


4. **Failed Test**: Your turn is over and play continues clockwise to the next player.
5. **Successful Test**: Place the Animal Token on the score board that was tested on the highest available point value on the score board for that animal.

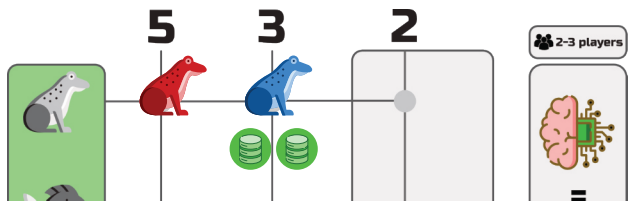
All players that pass a given animal in a round receive the same number of points.

- A round starts with the first player.
- Add Data Tokens next to player(s) animal figures so that all players have the same point value.
- Data Tokens on the scoreboard count as 1 point at the end of the game.





The blue player just successfully passed the frog test. The red (5 pts) player successfully tested the frog in a previous round, so the blue player takes the 2nd highest value spot for the frog (3 pts).

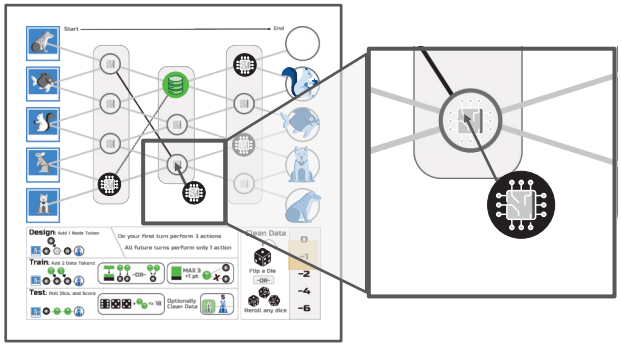


The blue player just successfully passed the frog test. The red (5 pts) player successfully tested the frog in the same round, so the blue player takes the 2nd hightest value spot for the frog (3 pts) + two data tokens (2 pts).

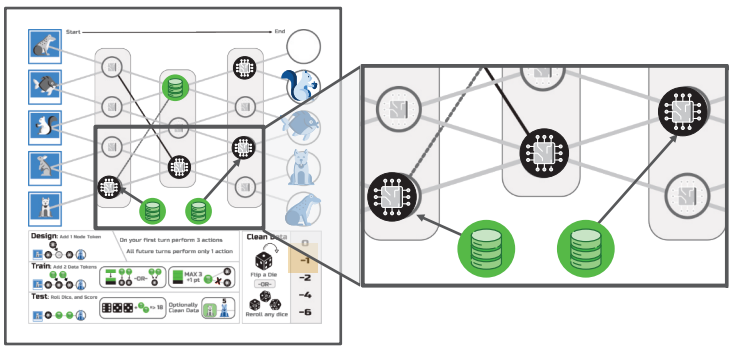
Example Game Play

Alex (blue player), is playing a 4 player game. This scenario is during the mid game, so some animals have already be successfully tested for all players. Alex wants to test the dog photo. He will spend 3 turns doing this.

Turn 1 (Design Model) - He needs to spend a turn completing the connection path to the dog.



Turn 2 (Train Model) - He could do a test now as there is a valid path for the dog photo, but the chances of success is low. He decides to train his model to prepare. This allows him to add two Data Tokens to his model.



Turn 3 (Test Model) - Roll 3 dice and evaluate.

12 + 2 = 14

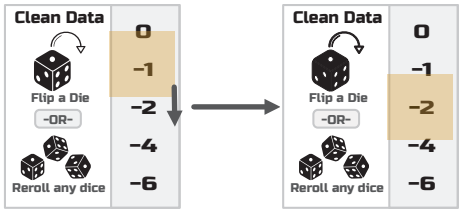
Failed Test

If Alex does not do anything the test would fail as the test status is below 18, and their turn is over.

Alex uses the data clean action to flip a die. This will allow him to pass the test this turn, but at a cost.

Flipping Die

He must slide the Data Clean Token down 1 space. Alex will receive a penalty at the end of the game for using the Clean Data action. Instead of receiving -1 point for using the Data Clean action in the earlier sequence, he will receive a penalty of -2 points.



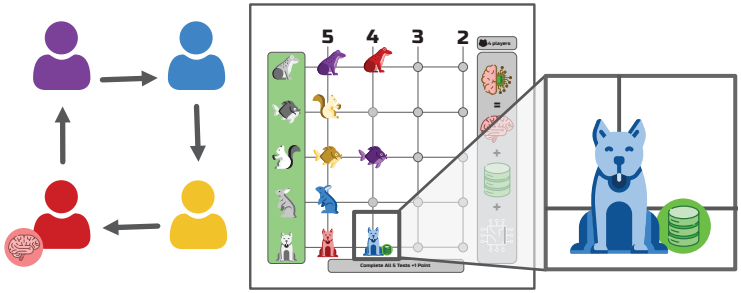
Now the test status is 18+. Alex successfully passed the test this turn.

Die flipped

17 + 2 = 19

Successful Test

The first player is the red player who also successfully passed the dog photo test this round. Both players receive the same number of points. The red player has the first score spot on the scoreboard for dog (5pts). Alex is in second place (4pts) and places 1 Data Token on the Score Board next to his Animal Token to indicate 1 additional point at the end of the game to give him a total of 5 points.



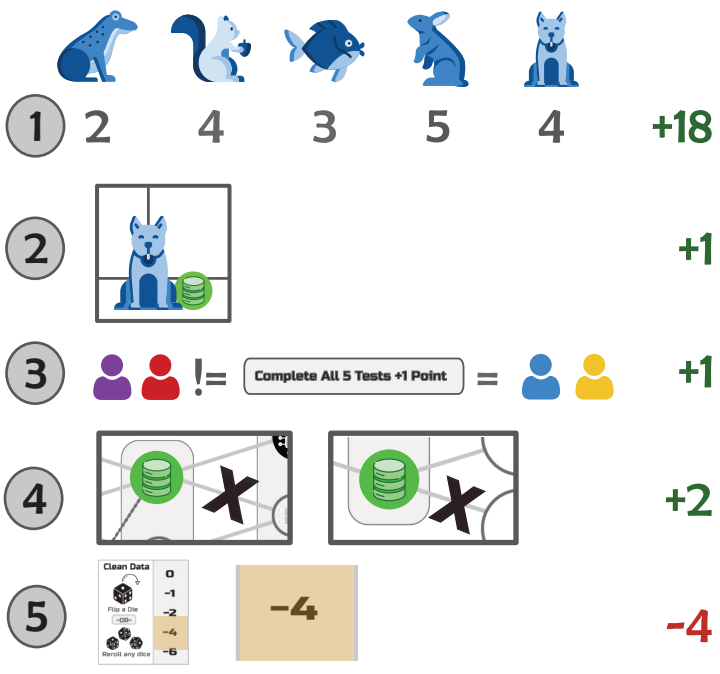
End Game Scoring

The highest score wins. How to score each player:

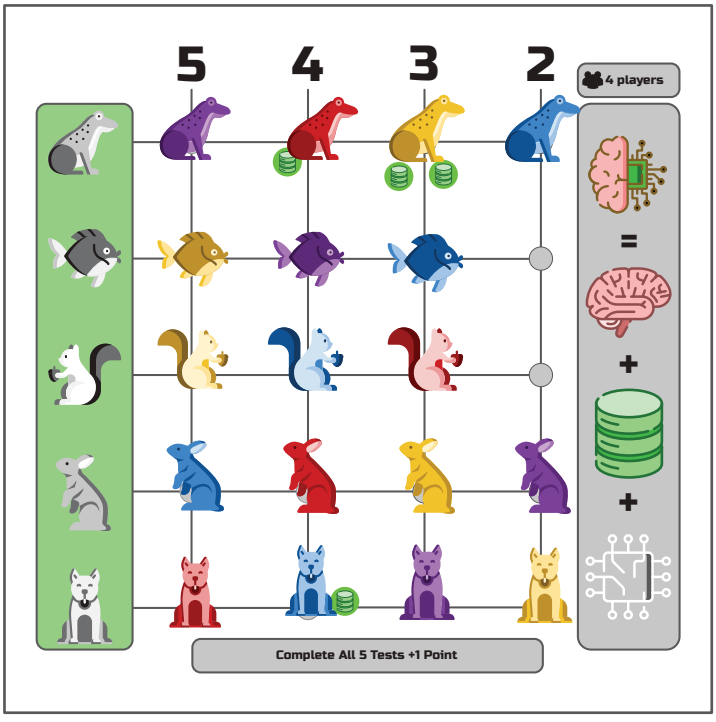
1. Each animal scores based on the column it is in. The first spot is 5 points, etc.
2. Any Data Tokens on the Score Board are +1 point.
3. 1 point if all 5 Animal Tokens are on the Score Board.
4. Each maxed node is +1 point on the Players Board.
5. Subtract the points the clean data token is over.

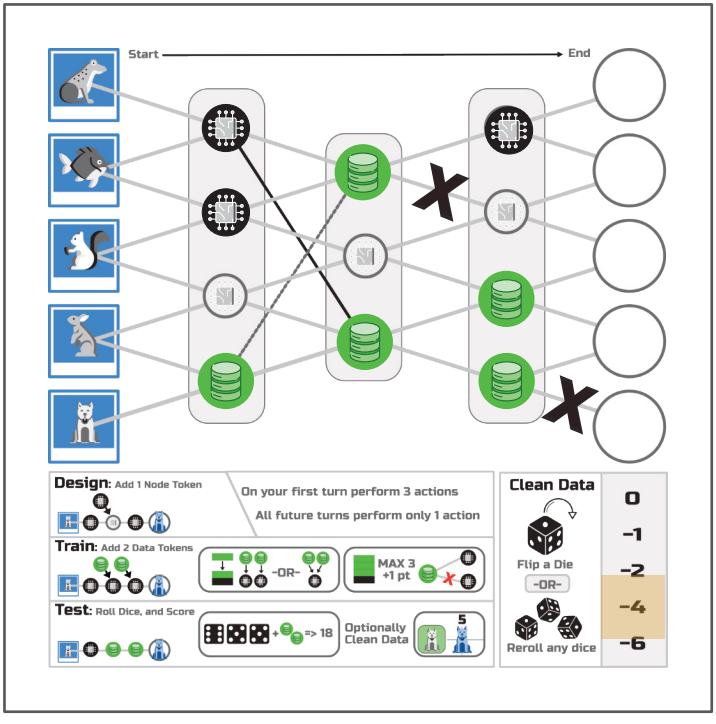
If there is a tie begin analyzing with the first scoring conditino to determine a winner. Continue down the list until a winner is identified. If there is still a tie after evaluating each scoring condition , then share the win.

We will use the blue player final state for this example.



Blue Players Final Score 18





Advance Mode

Now that you have mastered the basics, incorporate this rule to mimic real-world applications. Only apply this additional challenge after players are familiar with the standard rules.

When a player fails a test they must move one (and only one) Data Token on their board as the final step of their turn. The model "learns" from its mistakes. This process is called backpropagation, where the weights/bias (refer to page 8) are adjusted to improve future results.

Order of play:

1. Player fails a test.
2. (Optional) Use Clean Data action. Evaluate if the test status has changed and score animal.
3. Move a data token.

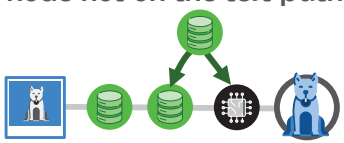
The moved Data Token does not impact the results of testing for the round the test failed, as it is only moved after the test status is finalized.

- Skip this step if there are no available Data Tokens.
- All normal Data Token placement rules apply, unless explicated stated here.
- A Data Token is not added to the Players' Board.

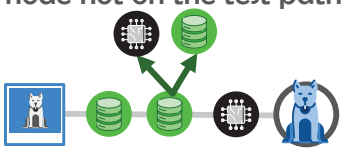
The path used in the failed test must be involved. The player performs one of the below options.

Transfer Data Token within the current test path

Receive a Data Token from a node not on the test path



Move a Data Token to a node not on the test path



Changing maxed out Node Tokens:

- If a Data Token is moved from a maxed out Node Token the Overfitting Token is removed.
- If a Data Token is moved to a Node Token and it becomes maxed out, an Overfitting Token must be added immediately.

Credits

Core Playtesters

Feifei Novack
Mike Krantz
Brian Kemp
Karen Kemp
Missy Wanstall

Editor

Sandy McVey

Art

Flaticon.com >>
- Freepik
- Roundicons Premium

Game Designer

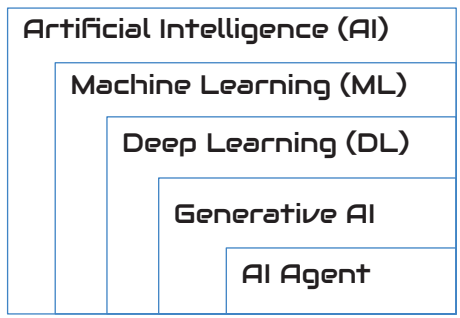
Michael Novack

Learn More

The goal of this section is to give more context to the learning objective of the game. This section is not required to understand to play the game.

People often interchangeably say AI and machine learning are the same thing. All machine learning is AI, but not all AI is machine learning.

AI >> Machine Learning >> Deep Learning >> GenAI >> AI Agent



In this game you create a neural network with multiple layers, which is a type of deep learning.

Deep learning is a type of machine learning (ML) that uses neural networks to learn from data. Neural networks are inspired by the human brain, and they can be used to solve a wide variety of problems, including image recognition, natural language processing, and speech recognition.

Neural networks are the basis for the modern machine learning advancements like Generative AI.

There are two key concepts that are important to understand the strengths and limitations of ML.

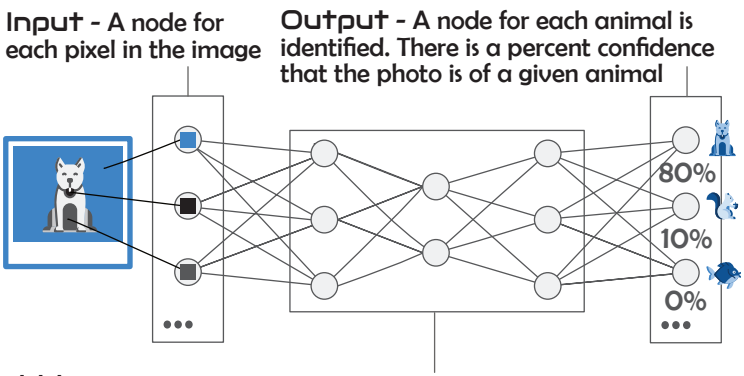
Probability Machine

ML will never be 100% accurate, and that is by design. This is a double-edge sword. It allows us to solve problems that are not possible to address with a rules-based/deterministic method. This is great when you can tolerate some variability in the results of the solution. We often stive for consistency in our systems. Do you want a chance that your mortgage payment goes to the wrong bank? It is about using the right tool for the right job, and ML is not always the best tool for the job.

Data is Destiny

ML needs high quantity and quality data to be effective. Data is the food of ML, as they say you are what you eat. If you only eat fast food, you might have a lot of fuel, but you will not be healthy in the long term. The same thing applies to ML. Getting lots of data is relatively easy. Obtaining quality data is diffcult. Ask yourself is everything on the internet 100% accurate. ML is limited by the data it is provided.

The following is an image of a neural network; notice the similarities to the Player Board. The diagram represents the connections in a neural network.



Each action in the game corresponds to real world actions you would perform to build a neural network.

Design

When you add a Node Token on your Player Board you are deciding the the size and structure of the hidden layers of your model. The more accurate name for a node is actually a neuron. The hidden layers decide how the input into the model will be interrupted and result in a given output. There is a balance between cost and accuracy related to the number of nodes and layers used in a model. The larger the hidden layer space the more accurate it becomes. But, also reflects higher cost it is to train and and added complexity to use.

Train

When you are training the model you are showing it labeled data, so that it can learn from it to make better predictions in the future. This is called supervised learning. Basically you are providing the model a picture of a dog and labeling that picture as a dog. The more samples it has the better it becomes. The training data determines the probability that a given animal will be identified in a picture. The wieght values between nodes and the bias values of each node is calculated to best fit the training data to maximize the accuracy of future predictions. If too much similar data is supplied a model can becom over fitted. If you give the model 80% dog photos it will be inclined to determine that future examples will be dog photos. That scenario is demonstrated in the game by blocked pathways when a given node receives too much training data.

Test

A successful model will accurately identify new images that were not included in the test data photos. Rolling the dice simulates providing a new photo for a specific animal and evaluating if the test is sucessful. As you train the model the likelihood of success increases, but is not guaranteed. This emulates the real world goal of 100% accuracy is no a realistic goal.