Explanation of Neural Network Architecture for Tic-Tac-Toe

1. Why 4 Layers?

The model has:

- 1 Input Layer (implicit): Defines the shape of the data (9 grid positions for Tic-Tac-Toe).

- 2 Hidden Layers: Allow the model to learn patterns from the input.

- 1 Output Layer: Produces the final prediction.

Reasons for this design:

- Two hidden layers are sufficient to solve a relatively simple problem like Tic-Tac-Toe.

- The first hidden layer learns basic patterns (e.g., if a row or column is close to winning).

- The second hidden layer refines and combines these patterns to detect winning scenarios.

2. Why 16 and 32 Neurons?

Hidden Layer 1 (16 neurons):

- Starts with a slightly larger number of neurons than the input features (9 grid positions).

- Captures simple patterns like rows, columns, or diagonals close to completion.

Hidden Layer 2 (32 neurons):

- Increases capacity to learn more complex relationships (e.g., interactions between multiple rows/columns).

- Allows for deeper feature learning, enabling the model to identify more abstract winning scenarios.

3. Why Sigmoid Activation in the Output Layer?

The sigmoid activation function is used in binary classification tasks like this one, where the goal is to predict whether Player X wins or not (two possible outcomes).

Sigmoid Function:

The sigmoid function converts any input into a probability between 0 and 1 using the formula:

    sigmoid(x) = 1 / (1 + exp(-x))

Why not Softmax?

- Softmax is used for multi-class classification tasks with more than two categories.

- In this binary classification problem, sigmoid is sufficient to output a single probability value.

4. Summary of the Architecture

Input Layer:

- 9 neurons, one for each grid position.

Hidden Layer 1:

- 16 neurons, ReLU activation, learns basic patterns.

Hidden Layer 2:

- 32 neurons, ReLU activation, learns complex relationships.

Output Layer:

- 1 neuron, Sigmoid activation, outputs a probability of Player X winning.

5. Further Reading

Here are some resources to learn more about designing neural networks:

- Deep Learning Book by Ian Goodfellow: https://www.deeplearningbook.org/

- Google AI Guidelines: https://developers.google.com/machine-learning/guides/neural-network-design

- Keras Activation Functions: https://keras.io/api/layers/activations/

- Stanford CS231n Course: http://cs231n.stanford.edu/