Neural Networks are computational models inspired by the human brain's interconnected neuron structure. They consist of layers of interconnected nodes that process data and learn patterns. Some of the Primary components are:
Neurons
Layers
Activation Functions
Weights and Biases
These components collectively enable neural networks to model intricate relationships in data, making them powerful for tasks like classification.

Task 1.

I created a neural network with one hidden layer and experimented with different activation functions: ReLU, Sigmoid, and Tanh.

Observations:

Sigmoid: It maps inputs to a range between 0 and 1, making it suitable for probabilistic interpretations. However, it can suffer from vanishing gradients, slowing down learning in deeper networks. The network Sigmoid activation showed slower convergence and struggled with the more complex patterns.

Tanh: Inputs a range between -1 and 1, centering the data and often leading to better convergence than Sigmoid. The network with Tanh activation performed better than Sigmoid but was still worse than ReLU.

Activation functions determine a neuron's output by adding non-linearity, enabling the network to learn more complex patterns. The activation function impacts the network's ability to converge.

Task 2.

I varied the number of neurons in the hidden layer and changed the number of hidden layers toi see their effects on performance.

Increasing Neurons in a single layer: Adding more neurons allowed the network to learn more complex patterns, reducing training loss. But beyond a certain point additional neurons led to overfitting.

Adding hidden layers: Adding more hidden layers enabled the network to model bigger patterns in the data. While this made the performance better to a point, too many layers resulted in increased computational complexity and potential overfitting.

Task 3.

Adjusted the learning rate to observe its impact on networks convergence speed and accuracy.

High learning rate: It converges quickly but often overshot optimal solutions leading to instability and higher training loss.

Low learning rate: The network converged more slowly, requiring more epochs to reach optimal solutions, but it achieved a lower training loss and more accurate results.

The learning rate controls the step size during optimization. A high learning rate can cause the model to miss optimal solutions, while a low learning rate ensures stable convergence but requires more iterations. Selecting an appropriate learning rate is crucial.

Task 4.

I introduced noise to the data using the “Noise” slider to see its effect on the network's generalization ability.

Low Noise: The network learned the pattern quickly, achieving low training and test loss.

High Noise: The network struggled to distinguish signal from noise, leading to higher training and test loss and reduced generalization.

Data noise refers to random variations or errors in the data. High noise levels can obscure underlying patterns, making it harder for the network to learn. Data preprocessing and noise reduction techniques are essential to improve model performance.

Task 5.

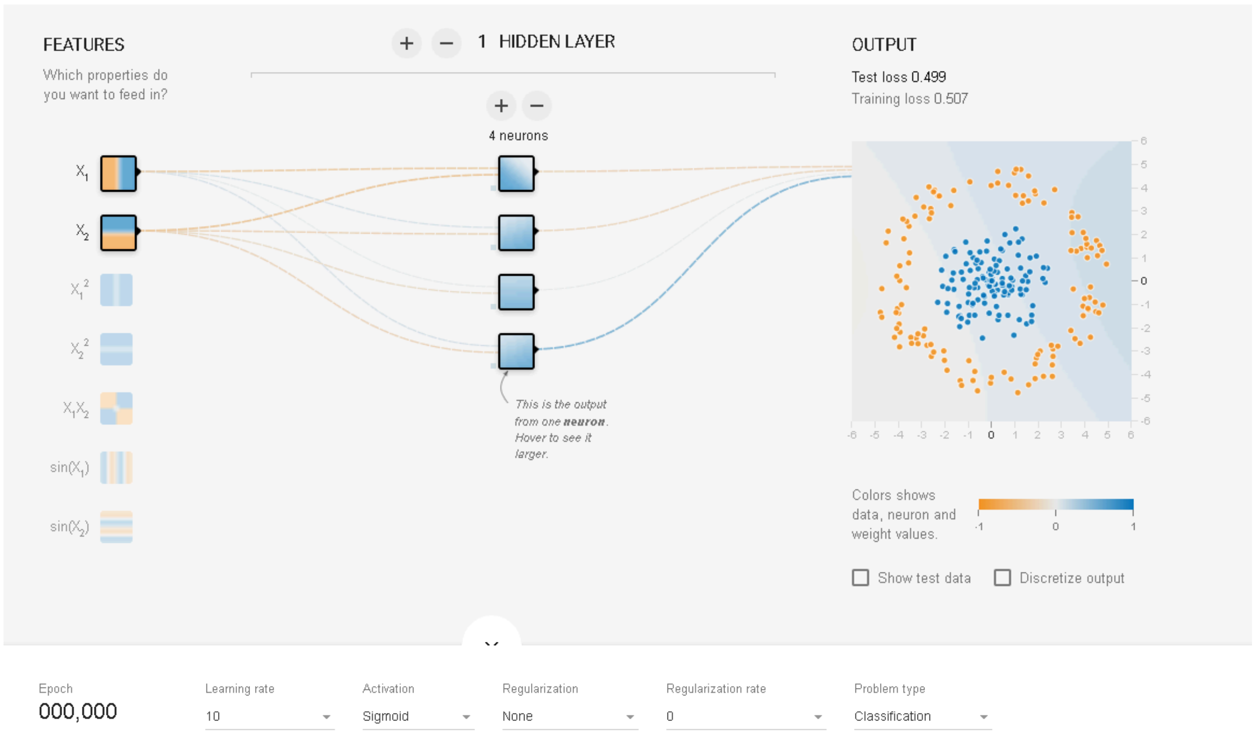
I explored different datasets available in TensorFlow Playground to analyze the network's performance.

Linear Dataset: The network easily learned achieving low training and test loss.

Non-linear Datasets: These datasets required more complex architectures for the networks to find the intricate pattern, bringing higher training and test loss.

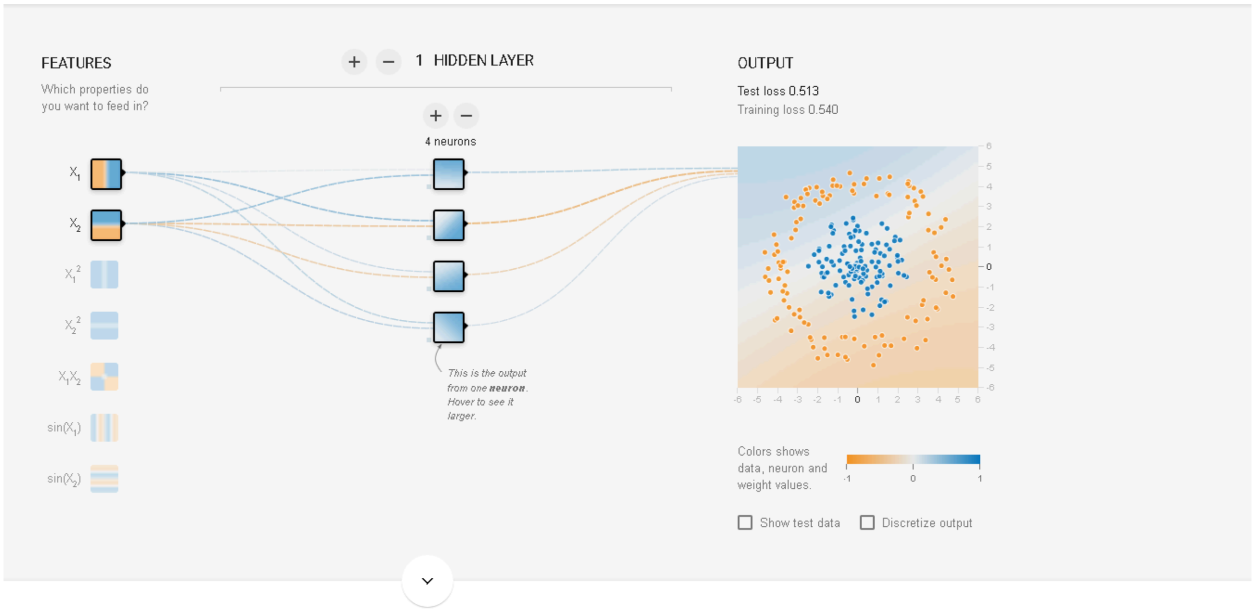
Conclusion.

This experience with TensorFlow Playground helped me gain more knowledge into how various parameters influence neural network performance and I learned about how activation functions have a critical role in introducing non-linearity and affecting the convergence.


FEATURES
Which properties do you want to feed in?
X1
X2
X1²
X2²
X1X2
sin(X1)
sin(X2)
1 HIDDEN LAYER
4 neurons
This is the output from one neuron. Hover to see it larger.
OUTPUT
Colors shows data, neuron and weight values.
-1
0
1
Show test data
Discretize output

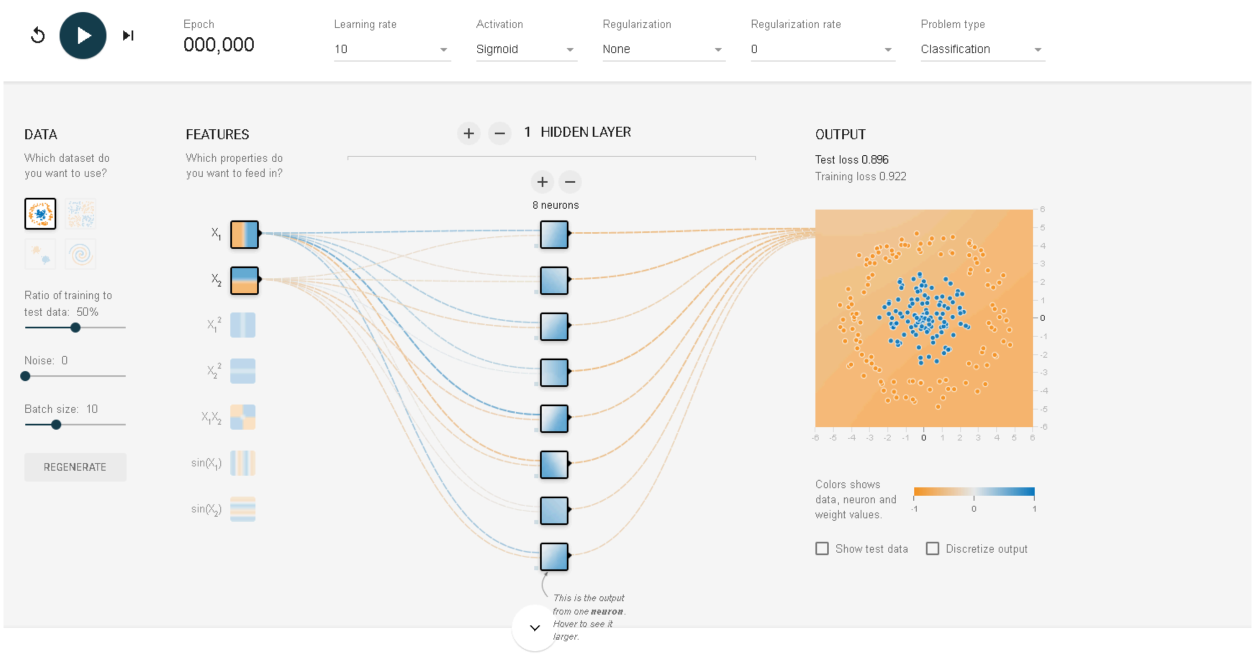
Test loss 0.499
Training loss 0.507

Epoch 000,000
Learning rate 10
Activation Sigmoid
Regularization None
Regularization rate 0
Problem type Classification


FEATURES
Which properties do you want to feed in?
X1
X2
X1²
X2²
X1X2
sin(X1)
sin(X2)
1 HIDDEN LAYER
4 neurons
This is the output from one neuron. Hover to see it larger.
OUTPUT
Colors shows data, neuron and weight values.
-1
0
1
Show test data
Discretize output

Test loss 0.513
Training loss 0.540

[illegible]



Epoch 000,000

Learning rate 10

Activation Sigmoid

Regularization None

Regularization rate 0

Problem type Classification

DATA

Which dataset do you want to use?

Ratio of training to test data: 50%

Noise: 0

Batch size: 10

REGENERATE

FEATURES

Which properties do you want to feed in?

1 HIDDEN LAYER

8 neurons

OUTPUT

Test loss 0.896

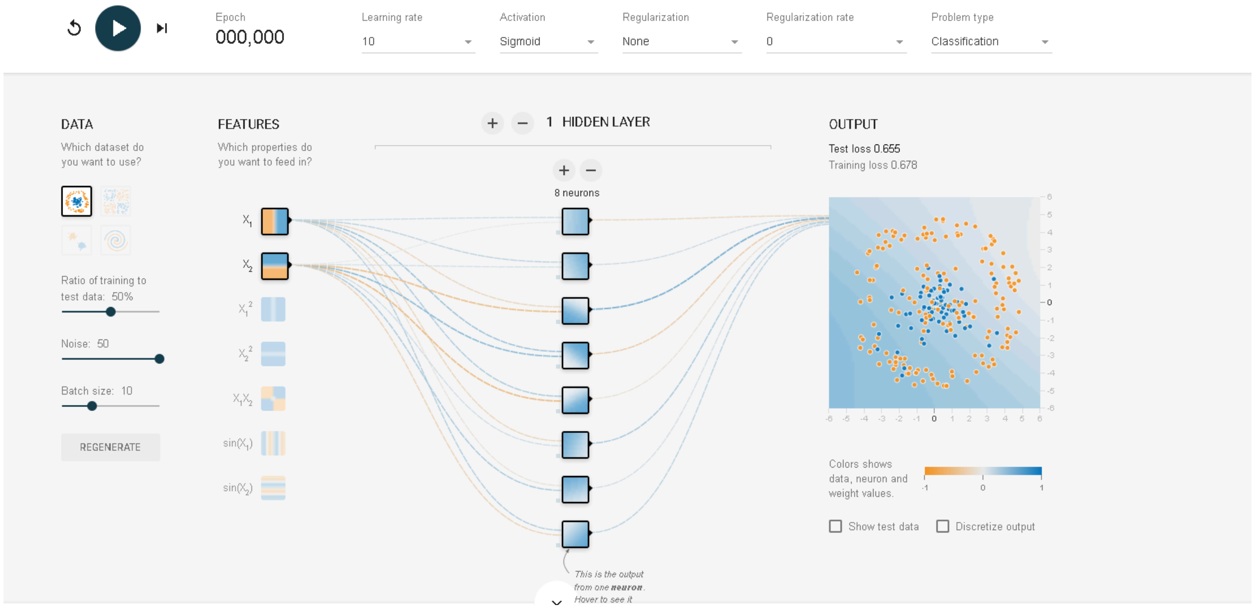
Training loss 0.922

Colors shows data, neuron and weight values.

Show test data

Discretize output

X_1
X_2
X_1^2
X_2^2
X_1X_2
$\sin(X_1)$
$\sin(X_2)$
This is the output from one **neuron**. Hover to see it larger.
-1
0
1



Epoch 000,000

Learning rate 10

Activation Sigmoid

Regularization None

Regularization rate 0

Problem type Classification

DATA

Which dataset do you want to use?

Ratio of training to test data: 50%

Noise: 50

Batch size: 10

REGENERATE

FEATURES

Which properties do you want to feed in?

1 HIDDEN LAYER

8 neurons

OUTPUT

Test loss 0.655

Training loss 0.678

Colors shows data, neuron and weight values.

Show test data

Discretize output