

A05 TensorFlow Playground Presentation

Binary Brains:

Ambalika

Favour

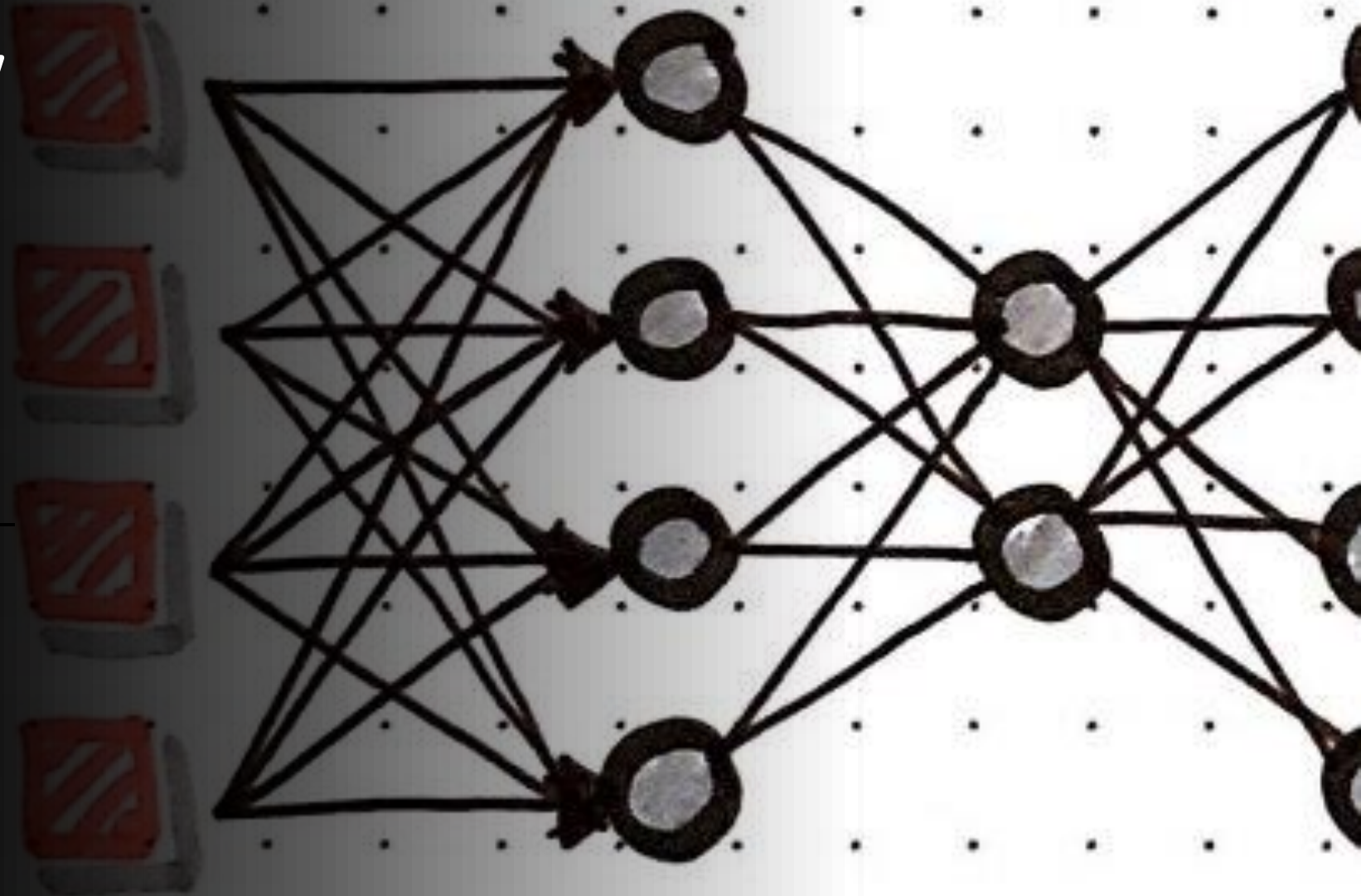
Misty

Joseph

Zaid

Inputs

Hidden Layer



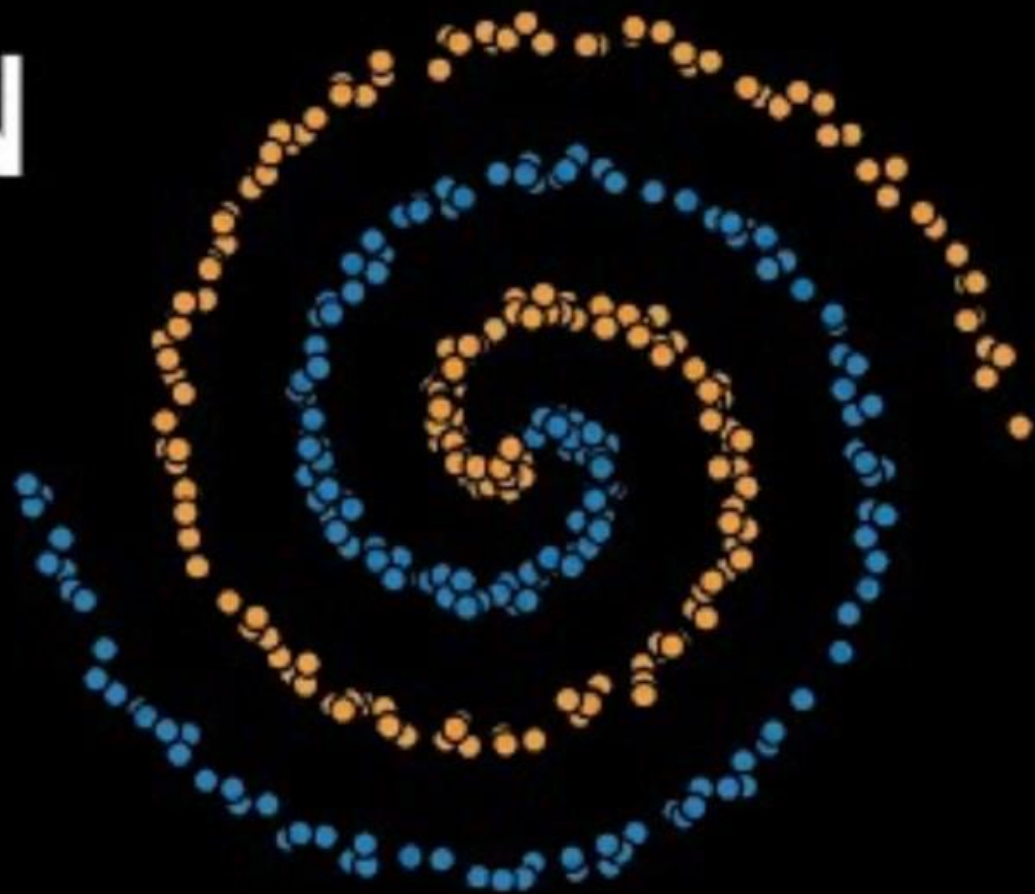
Purpose of the Exploration:

The exploration aims to understand how different parameters influence the behavior and performance of neural networks.

It provides practical insights into the impact of parameter settings on convergence, accuracy, and generalization of the models.

Understanding parameter dynamics is crucial for effectively designing and training neural networks for real-world applications.

**NEURAL NETWORKS
CAN LEARN
(ALMOST)
ANYTHING**



What is Tensor flow and how does it work?

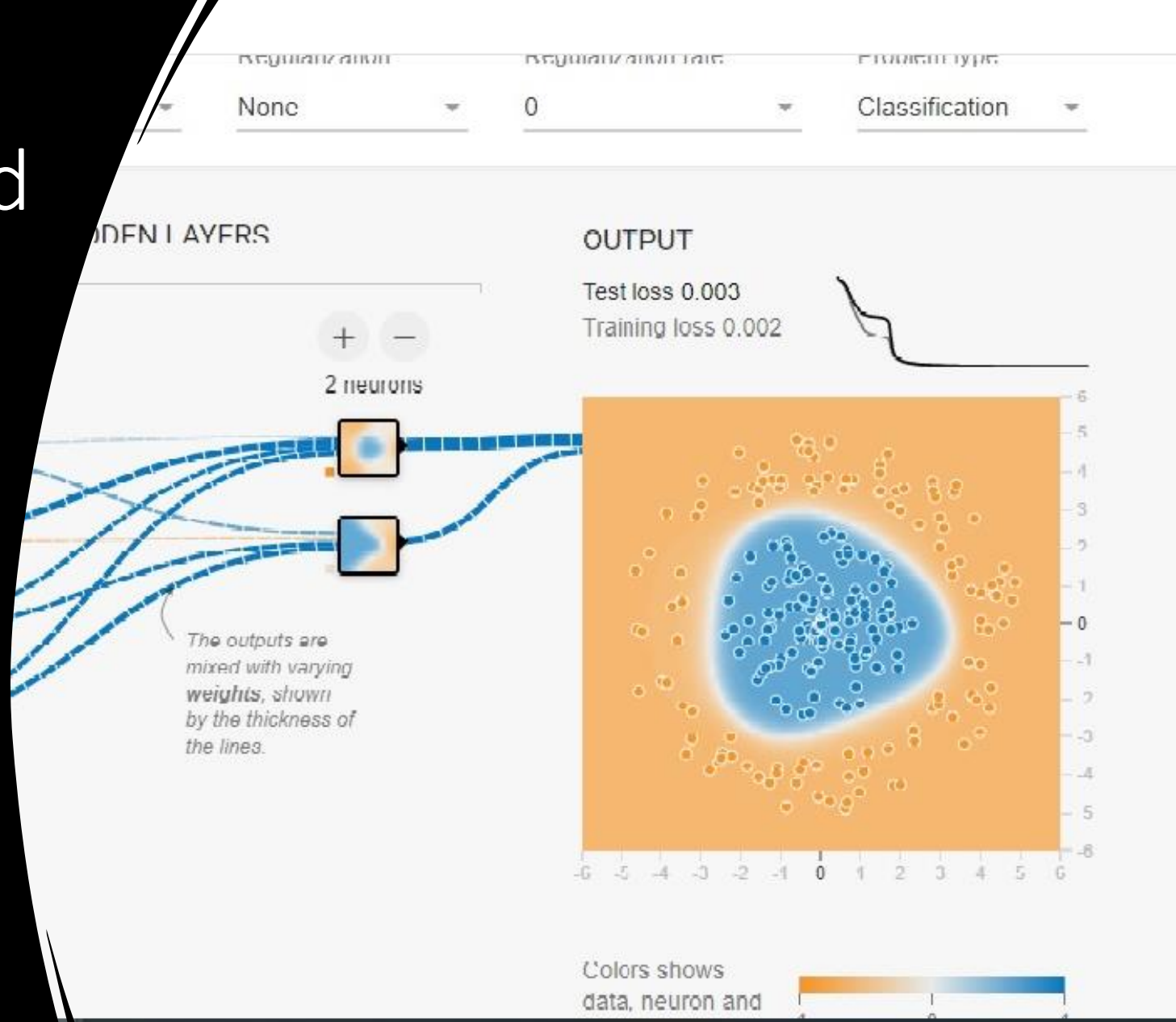
TensorFlow playground is made by the TensorFlow team at Google, and it is a great way of learning about deep neural networks. We can use it to explore and see how powerful these networks are.

The "run" and "pause" buttons control the training of the neural network. When you click "run," the epochs (training iterations) continue running until you stop them. The "pause" button stops the training.

Regularization: It is a technique that makes slight modifications to the learning algorithm such that the model generalizes better.

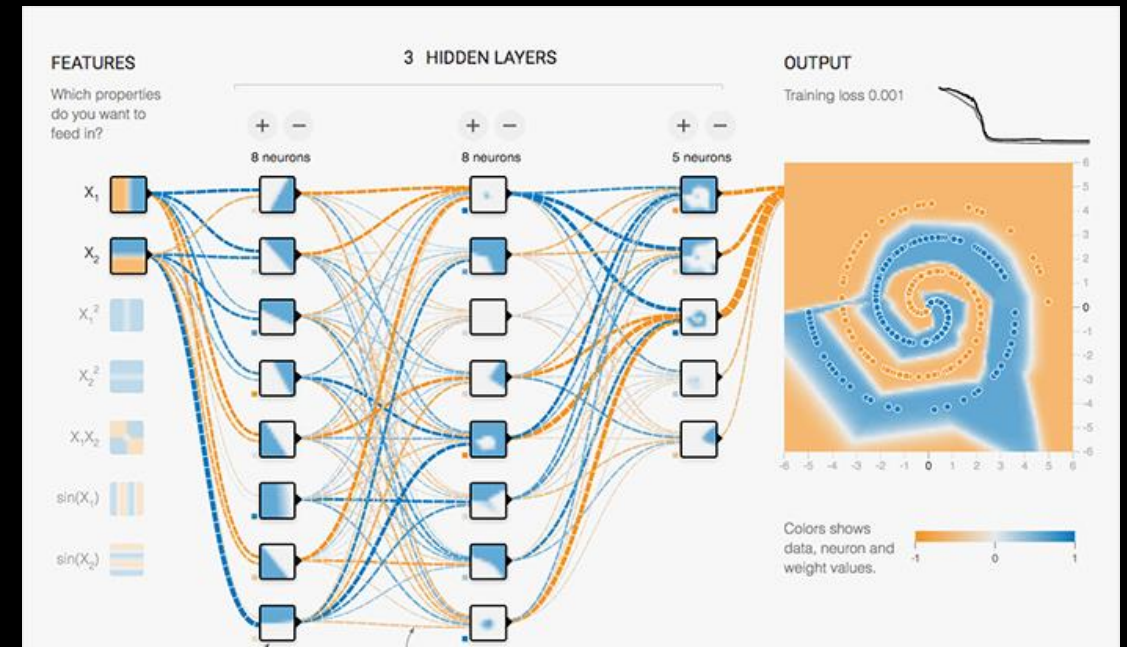
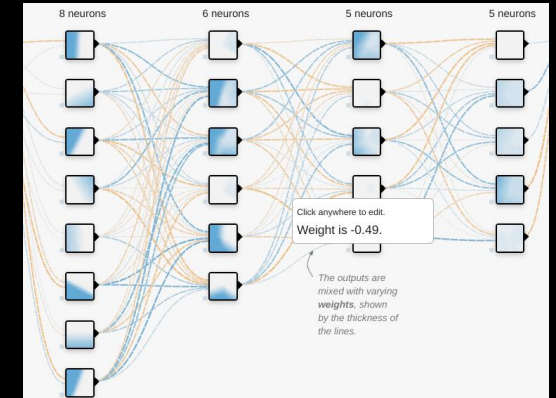
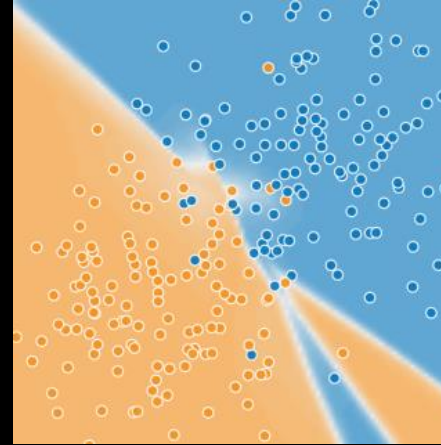
Regularization Rate: This option lets you set the regularization rate.

Problem Type: You can choose between classification or regression problems. We'll focus on classification for this demonstration.



TensorFlow Playground Experience

- The TensorFlow Playground Lab demonstrated that even simple changes, like altering activation functions or network architecture, can have varying impacts on model performance.
- Adjusting hyperparameters like learning rate underscored the delicate trade-off between convergence speed and accuracy.
- Introducing noise highlighted the network's ability to generalize but also the necessity of clean data for precise classification



Activation functions:

- Non-linear functions applied to neuron outputs, influencing the network's ability to model complex relationships in the data.
 - Activation: Activation functions improve the performance of neural networks. There are four options: tanh, ReLU, Sigmoid, and linear.
- ReLU showed fast convergence, while sigmoid offered smoother outputs. Each function had its trade-offs, emphasizing the importance of thoughtful choices.

Impact of Parameters on Performance:

- Varying the number of neurons can affect the model's capacity to learn, potentially leading to underfitting or overfitting.
- The learning rate significantly influences the convergence speed and the stability of the training process.
- Activation functions impact the non-linearity of the model, affecting its ability to approximate complex functions and avoid vanishing gradients.

Practical Implications:

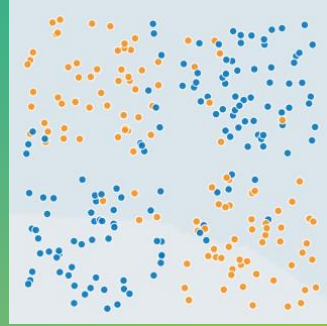
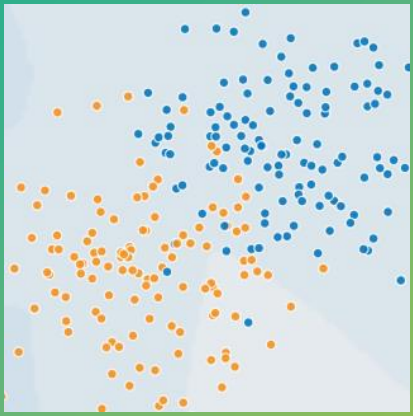
- Understanding parameter impacts helps in designing efficient neural network architectures tailored to specific tasks and datasets.
- Optimizing parameters can improve training efficiency, reduce computational costs, and enhance model generalization.
- Insights gained from parameter exploration can inform decisions in real-world applications, such as selecting appropriate architectures and fine-tuning hyperparameters.

Insights Gained:

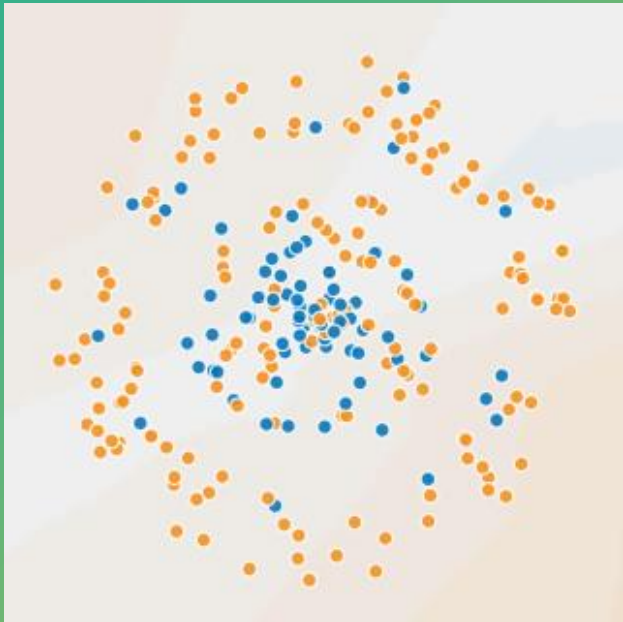
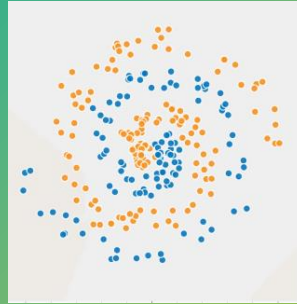
Experimentation with different parameter configurations provides valuable insights into the behavior and performance of neural networks. Iterative exploration fosters a deeper understanding of the trade-offs involved in model design and optimization. Insights gained from hands-on exploration contribute to the development of effective strategies for building and deploying neural network models.



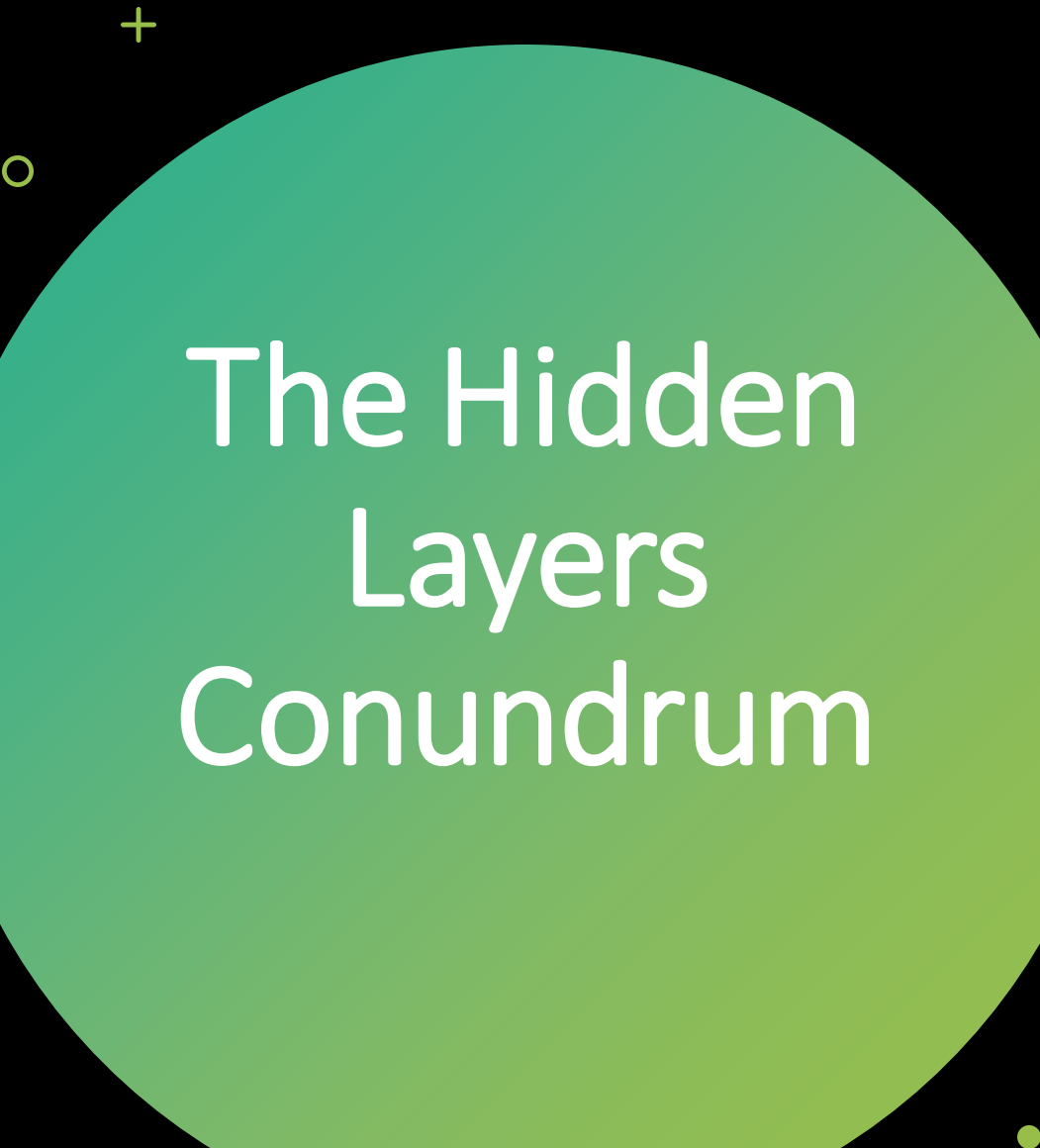
Lab Learnings



+



- Increasing complexity, such as altering the number of hidden layers and neurons, does not always lead to improved performance, emphasizing the nuanced relationship between model architecture and outcomes.
- The lab also emphasized the delicate balance in choosing the learning rate, underscoring the importance of hyperparameter tuning.
- Lastly, as noise can impact the model's ability to generalize and lead to lower accuracy, the crucial role of clean data for accurate categorization cannot be overstated.




The Hidden Layers Conundrum

We experimented with hidden layers and neurons. Surprisingly, the complexity didn't always lead to better results. Fewer neurons meant slower convergence but simpler models. More neurons speed things up but risk overfitting.

Number of neurons in the hidden layer(s):

Determines the capacity of the model to capture complex patterns



The Learning Rates

Learning Rate: This determines how quickly the neural network adjusts to fit the output correctly. You can change it to see how it affects learning.

Adjusting the learning rate was like fine-tuning an instrument. High rates raced toward convergence but sometimes overshoot the mark. Low rates ensured stability but at the cost of sluggish progress. I realized that finding the sweet spot required patience and intuition. It Controls the size of the steps taken during gradient descent optimization and affects the convergence speed.



An abstract graphic on the left side of the slide, featuring a complex network of white lines and dots on a dark blue background, resembling a data network or a molecular structure.

Noise and Clean Data

Introducing noise into the data revealed the network's ability to generalize. But it also underscored the need for pristine data. Noise impacted accuracy, reminding me that garbage in equals garbage out. Clean data was the bedrock of reliable classification.

DATA

Which dataset do you want to use?



Ratio of training to test data: 50%



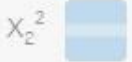
Noise: 0



Batch size: 10

FEATURES

Which properties do you want to feed in?



3 HIDDEN LAYERS

+ -

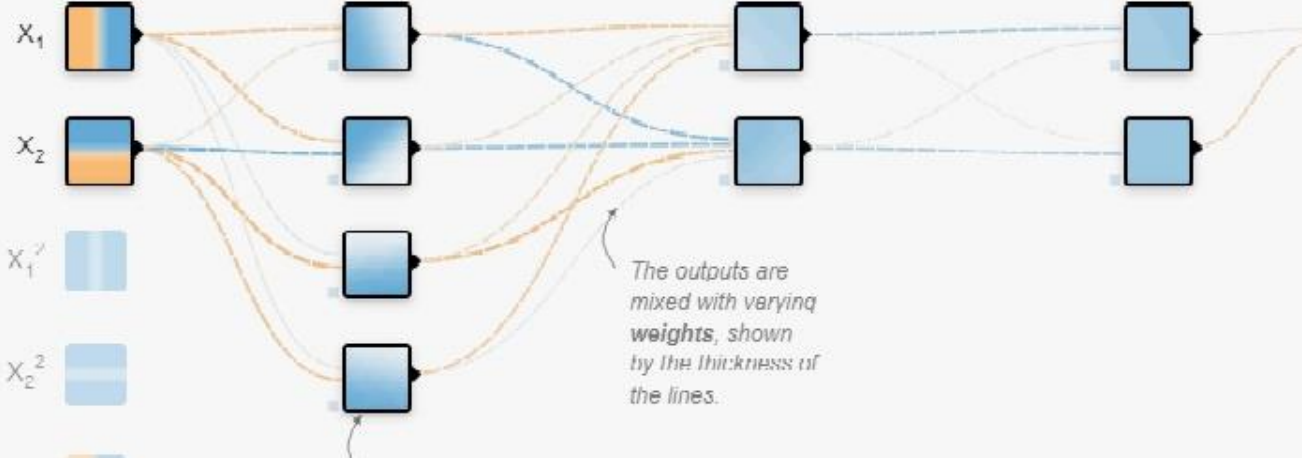
1 neurons

+ -

2 neurons

+ -

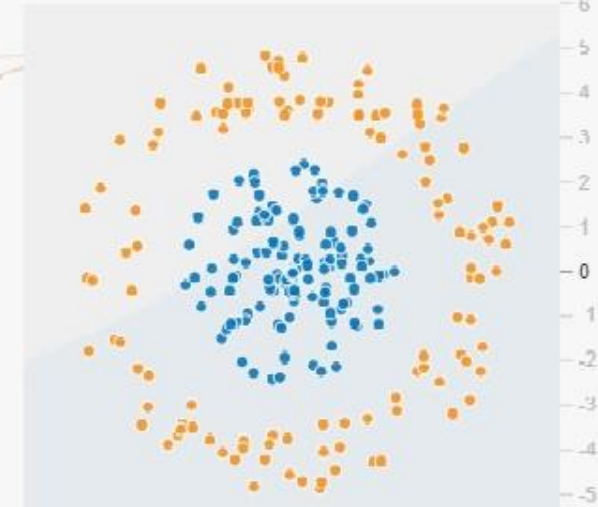
2 neurons



OUTPUT

Test loss 0.502

Training loss 0.500

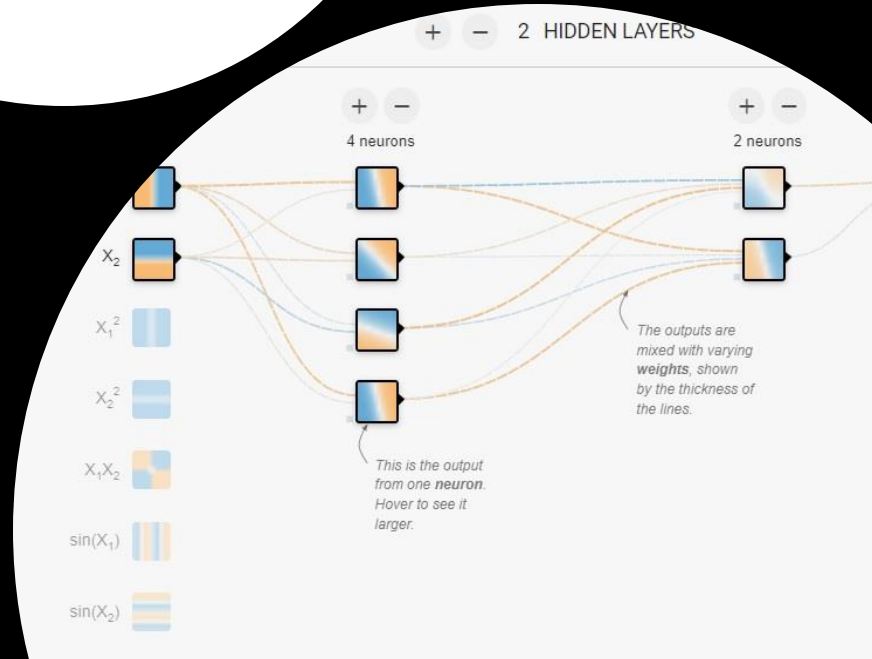
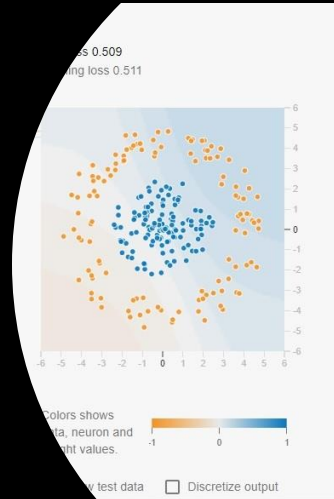


Dataset Types

- We have four datasets of increasing difficulty: Circular, Exclusive OR, Gaussian, and Spiral. We can choose to split the dataset into training and testing sets, with a default 50% split for each. If we want, we can add up to 50 noise points to the dataset. There's also a button to set the batch size for each epoch, and a "regenerate" button to generate new data.
- Throughout the visualization, orange and blue colors are used. Orange typically represents negative values, while blue represents positive values. The data points are initially colored orange or blue, indicating whether they belong to the positive or negative class

Neural Network Architecture

- By default, we have a 1:2:1 neural network architecture, which means 1 layer on inputs with 2 nodes, 2 hidden layers with the first layer having 4 nodes, 2 nodes in 2nd hidden layer, and a final output layer.
- In the hidden layers, the lines are colored by the weights of the connections between neurons. Blue shows a positive weight, which means the network is using that output of the neuron as given. An orange line shows that the network is assigning a negative weight.
- In the output layer, the dots are colored orange or blue, depending on their original values. The background color shows what the network is predicting for a particular area. The intensity of the color shows how confident that prediction is.



References:

- TensorflowPlayground: <https://playground.tensorflow.org/tinkerwithneuralnetwork>
- "Why neural networks can learn (almost) anything": <https://www.youtube.com/watch?v=0QczhVg5HaI>
- "Understanding neural networks with Tensorflow playground": <https://cloud.google.com/blog/products/ai-machine-learning/understanding-neural-networks-with-tensorflow-playground>
- Explained: Neural Networks: <https://news.mit.edu/2017/explained-neural-networks-deep-learning-0414>
- Neural Networks Made Fun With TensorFlow Playground: <https://towardsdatascience.com/neural-networks-made-fun-with-tensorflow-playground-4e681a0c4529>
- An outside opinion on Tensorflow playground: A Neural Network Playground. Build, train, and gain intuition for... | by Joshua Pickard | GeekCulture | Medium