A06 Report: Understanding Neural Networks Through TensorFlow Playground

ITAI-1378 Computer Vision in Artificial Intelligence

Professor Patricia McManus

Group 4:

Hisham D Macaraya

Katherine Stanton

Dallas Foster

Kaire Campomanes

Elijah Ghaya

Introduction to Neural Networks and Their Components

Neural networks are computational models inspired by the human brain. They are used for a wide range of tasks, including classification, regression, and pattern recognition. A neural network consists of several layers of interconnected neurons or nodes that pass information through weighted connections. Key components of a neural network include:

- **Neurons**: The fundamental processing units that receive an input, apply a weight, and pass it through an activation function.
- **Layers**: Neural networks have multiple layers—input layers, hidden layers, and output layers. The hidden layers are where most of the learning takes place as data gets transformed into different feature representations.
- **Activation Functions**: These introduce non-linearity into the model, allowing the network to learn complex relationships. Examples include ReLU, sigmoid, and tanh.

Neural networks are significant because they can learn from data, make predictions, and adapt to new information. They can also model complex functions and relationships, making them suitable for various real-world applications, such as image recognition, natural language processing, and autonomous driving (Goodfellow et al., 2016).

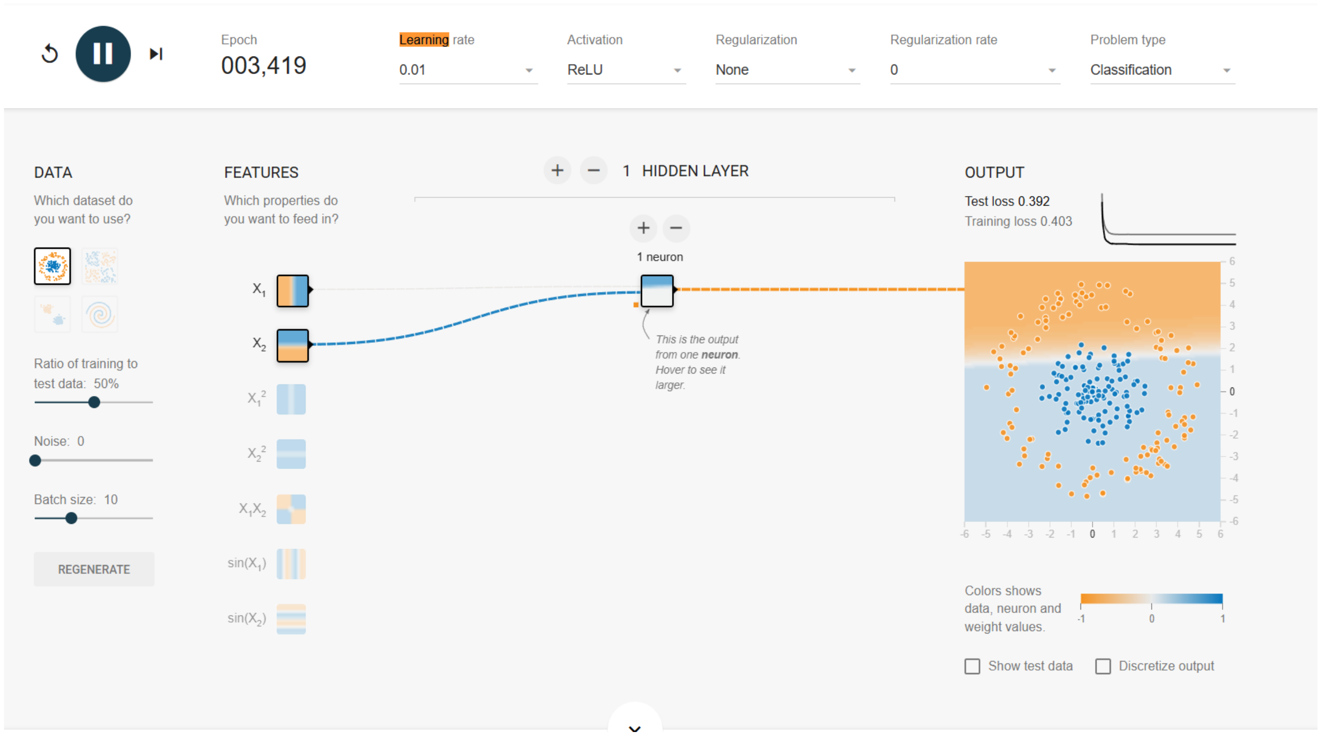
Tasks and Observations

These simulations/tasks are made on the TensorFlow Playground website: **https://playground.tensorflow.org**

This Photo by Unknown Author is licensed under CC BY

Task 1: Activation Functions

- **Configuration**: The network consisted of 1 hidden layer with 1 neuron, and different activation functions—ReLU and sigmoid—were tested.
- **Observations**: With ReLU (Fig. 1), the model struggled with complex data due to the architecture's insufficient complexity. Sigmoid showed similar behavior, indicating a lack of non-linear capacity with just one neuron.



Epoch
003,419

Learning rate
0.01

Activation
ReLU

Regularization
None

Regularization rate
0

Problem type
Classification

Ratio of training to test data: 50%

Noise: 0

Batch size: 10

Test loss 0.392
Training loss 0.403

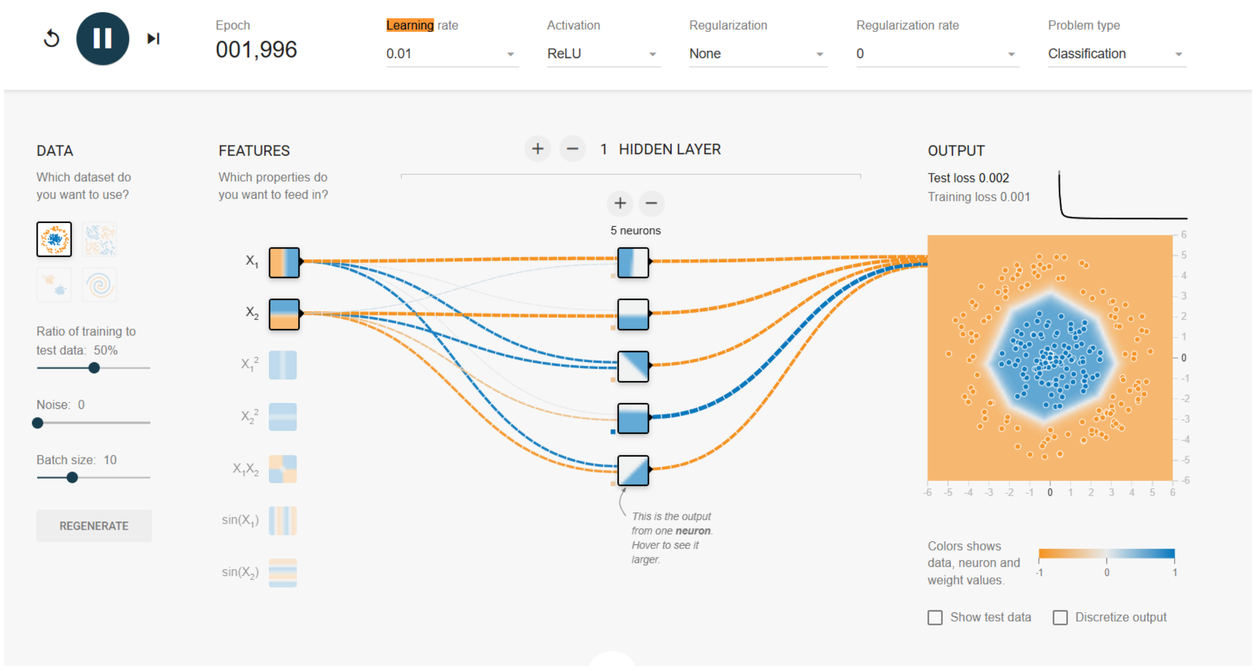
DATA
Which dataset do you want to use?
REGENERATE
FEATURES
Which properties do you want to feed in?
1 HIDDEN LAYER
1 neuron
This is the output from one neuron. Hover to see it larger.
OUTPUT
Colors shows data, neuron and weight values.
Show test data
Discretize output

Figure 1: Screenshot of the model using ReLU activation with a test loss of 0.500.

Activation functions are crucial as they add the non-linearity needed to model complex relationships (Nair & Hinton, 2010).

Task 2: Hidden Layer Neurons

- **Configuration**: The number of neurons in the hidden layer was increased to 5. The ReLU activation function was retained.
- **Observations**: Adding more neurons resulted in a significant decrease in both training loss and test loss (Fig. 2), showing that increasing the number of neurons improves the model's ability to capture complex patterns.


Epoch 001,996

Learning rate 0.01

Activation ReLU

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

FEATURES

Which properties do you want to feed in?

1 HIDDEN LAYER

5 neurons

This is the output from one **neuron**. Hover to see it larger.

OUTPUT

Test loss 0.002

Training loss 0.001

Colors shows data, neuron and weight values.

Show test data

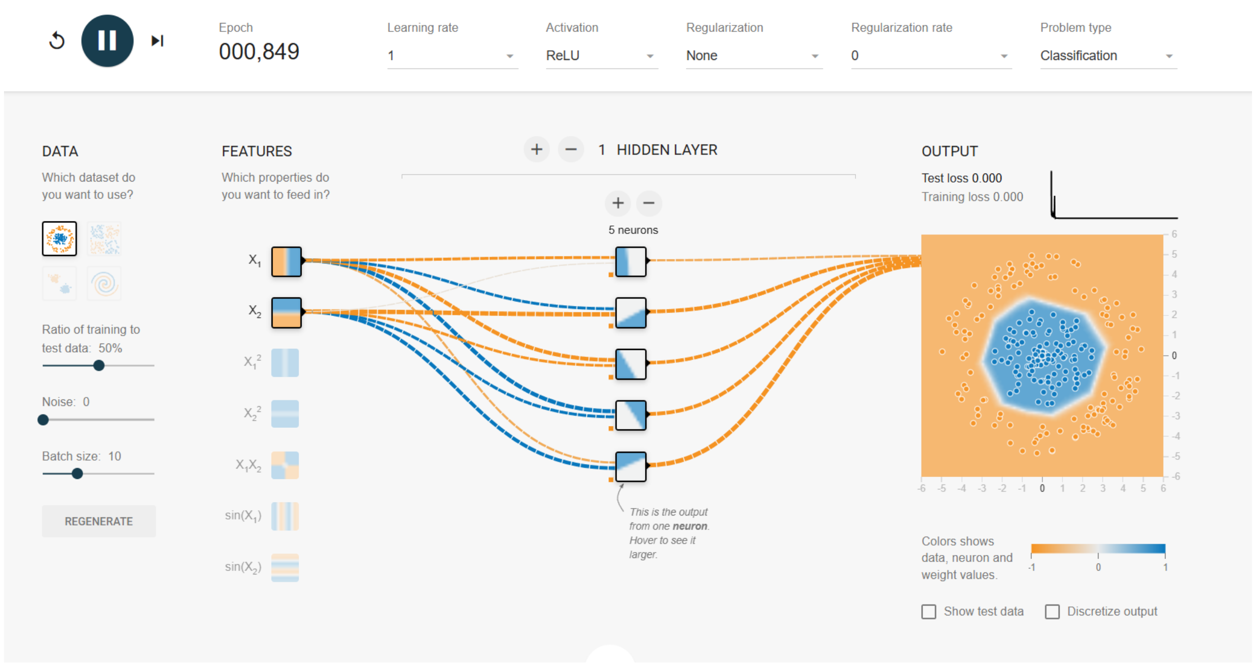
Discretize output

Figure 2: Screenshot showing the impact of increasing neurons to 5, reducing test loss to 0.002.

Increasing the number of neurons allows the network to learn more nuanced features and form more complex decision boundaries (Goodfellow et al., 2016).

Task 3: Learning Rate

- **Configuration**: The learning rate was increased to 1.
- **Observations**: With 5 neurons in the hidden layer, a learning rate of 1 led to fast convergence (Fig. 3), achieving 0.000 test loss. However, a high learning rate can sometimes cause the model to overshoot optimal solutions.


Epoch 000,849
Learning rate 1
Activation ReLU
Regularization None
Regularization rate 0
Problem type Classification

Ratio of training to test data: 50%
Noise: 0
Batch size: 10

Test loss 0.000
Training loss 0.000

This is the output from one **neuron**. Hover to see it larger.

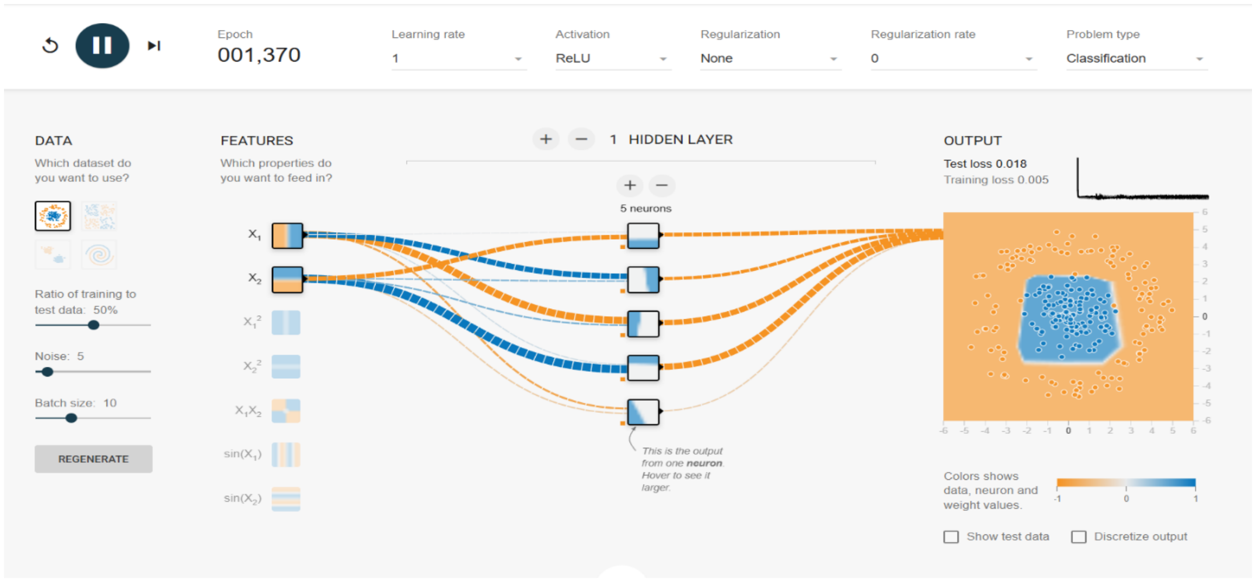
Colors shows data, neuron and weight values.

Figure 3: Screenshot of the model with a learning rate of 1 and test loss of 0.000.

A high learning rate speeds up training, but care must be taken to avoid instability (Kingma & Ba, 2014).

Task 4: Data Noise

- **Configuration**: Noise was added to the data with a noise level of **5**.
- **Observations**: The network's performance decreased, resulting in a **test loss of 0.018** (Fig. 4). Noise made it harder for the model to learn the correct patterns, leading to misclassifications.



Epoch
001,370

Learning rate
1

Activation
ReLU

Regularization
None

Regularization rate
0

Problem type
Classification

DATA
Which dataset do you want to use?

Ratio of training to test data: 50%

Noise: 5

Batch size: 10

FEATURES
Which properties do you want to feed in?

1 HIDDEN LAYER

5 neurons

This is the output from one **neuron**. Hover to see it larger.

OUTPUT
Test loss 0.018
Training loss 0.005

Colors shows data, neuron and weight values.

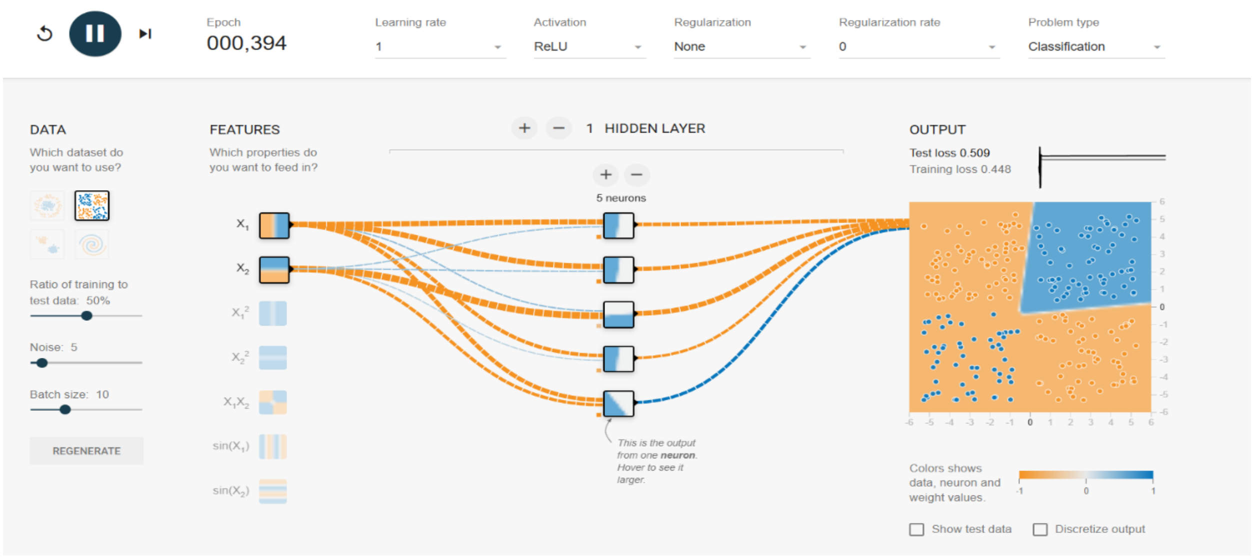
Figure 4: Screenshot showing the impact of noise with test loss at 0.018.

Noise can introduce irrelevant variations that reduce the model's ability to generalize (Dietterich, 1995).

Task 5: Dataset Exploration

a) XOR Dataset

- **Configuration**: **1 hidden layer** with **5 neurons**, **ReLU activation**.
- **Observations**: The model struggled with the **XOR** dataset, achieving a high **test loss of 0.509** (Fig. 5). This demonstrates the limitations of using linear boundaries for non-linearly separable data.


Epoch 000,394
Learning rate 1
Activation ReLU
Regularization None
Regularization rate 0
Problem type Classification

Ratio of training to test data: 50%
Noise: 5
Batch size: 10

Test loss 0.509
Training loss 0.448

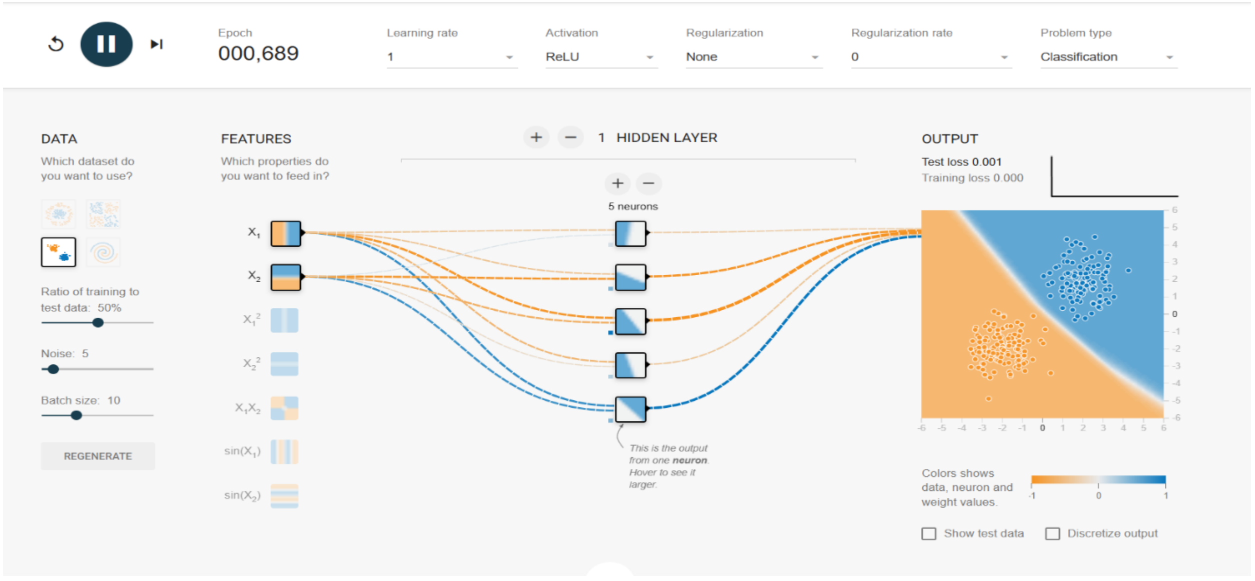
DATA
Which dataset do you want to use?
REGENERATE
FEATURES
Which properties do you want to feed in?
1 HIDDEN LAYER
5 neurons
This is the output from one neuron. Hover to see it larger.
OUTPUT
Colors shows data, neuron and weight values.
Show test data
Discretize output

Figure 5: Screenshot of the XOR dataset showing poor decision boundaries.

The XOR dataset illustrates the need for more complex architectures to handle non-linear relationships (Minsky & Papert, 1969).

b) Gaussian Dataset

- **Configuration**: Same architecture with **5 neurons**.
- **Observations**: The model performed well with the **Gaussian** dataset, achieving **0.001** test loss (Fig. 6). The Gaussian clusters were easily separable with the given configuration.


Epoch 000,689
Learning rate 1
Activation ReLU
Regularization None
Regularization rate 0
Problem type Classification

Ratio of training to test data: 50%
Noise: 5
Batch size: 10

Test loss 0.001
Training loss 0.000

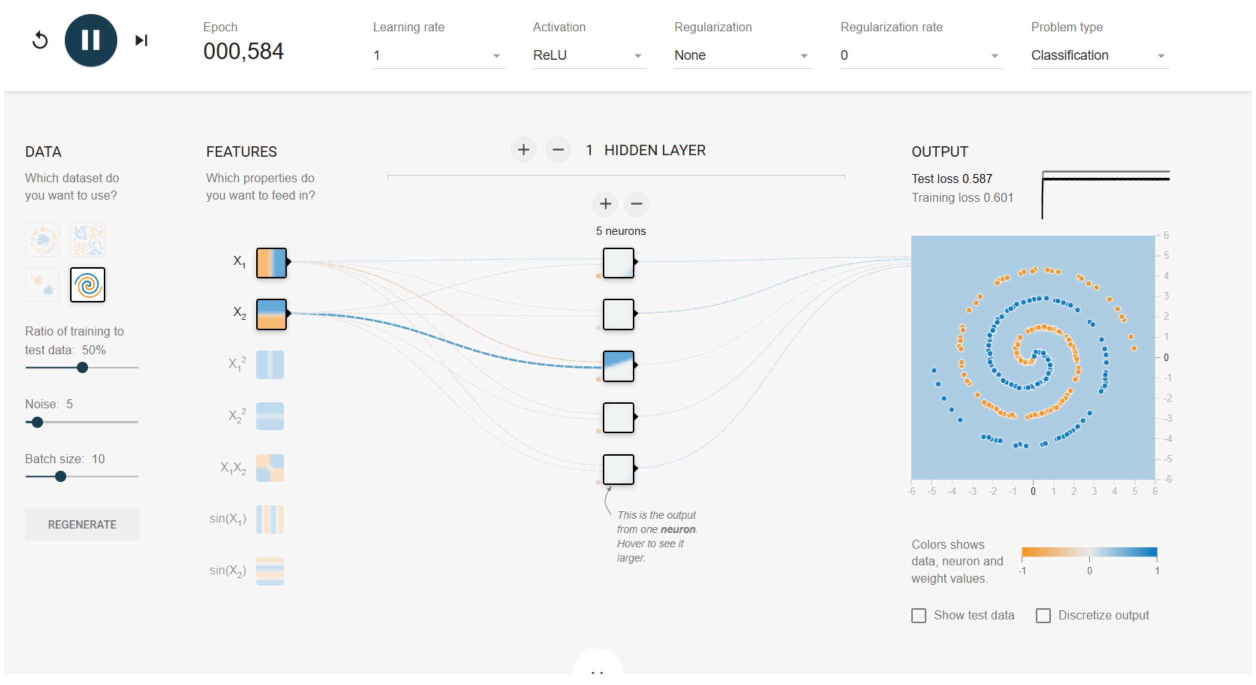
DATA
Which dataset do you want to use?
REGENERATE
FEATURES
Which properties do you want to feed in?
1 HIDDEN LAYER
5 neurons
This is the output from one neuron. Hover to see it larger.
OUTPUT
Colors shows data, neuron and weight values.
Show test data
Discretize output

Figure 6: Screenshot of the Gaussian dataset showing an accurate decision boundary.

Proper dataset selection is crucial, as it should match the complexity of the model's architecture (LeCun et al., 2015).

c) Spiral Dataset

- **Configuration**: Same architecture with **5 neurons**.
- **Observations**: The model was unable to classify the **spiral dataset** effectively, resulting in a **test loss of 0.587** (Fig. 7). The spiral structure requires more complex, non-linear decision boundaries.


Epoch 000,584
Learning rate 1
Activation ReLU
Regularization None
Regularization rate 0
Problem type Classification

Ratio of training to test data: 50%
Noise: 5
Batch size: 10

1 HIDDEN LAYER
5 neurons

Test loss 0.587
Training loss 0.601

DATA
Which dataset do you want to use?
REGENERATE
FEATURES
Which properties do you want to feed in?
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.
OUTPUT
Colors shows data, neuron and weight values.
-1
0
1
Show test data
Discretize output

Figure 7: Screenshot of the spiral dataset showing poor performance.

Complex datasets like the spiral dataset require deeper networks or more advanced architectures for effective classification (Goodfellow et al., 2016).

Practical Implications of Findings

1. **Activation Functions**: Understanding activation functions and their impact on non-linearity is key to building models capable of learning complex patterns. For example, ReLU is effective for deep networks as it mitigates the vanishing gradient problem (Nair & Hinton, 2010).
2. **Neuron Count**: Increasing the number of neurons enhances the network's ability to learn detailed features, improving performance on complex datasets. However, overfitting may occur if too many neurons are used, especially for simple datasets (Goodfellow et al., 2016).
3. **Learning Rate**: Adjusting the learning rate is critical for efficient training. A high learning rate can speed up convergence but risks overshooting the optimal point. This is essential in scenarios like hyperparameter tuning for real-world machine learning applications (Kingma & Ba, 2014).
4. **Data Noise**: Noise complicates the learning process, often requiring regularization techniques to ensure that the model generalizes well. This is relevant in applications involving noisy data, such as speech recognition or sensor data analysis (Dietterich, 1995).
5. **Dataset Selection**: The choice of dataset influences the model's architecture. Datasets like Gaussian were easy to classify, whereas XOR and spiral required more complexity, demonstrating the need for adaptive models depending on data characteristics (LeCun et al., 2015).

Conclusion

This hands-on exploration of neural networks using TensorFlow Playground has provided a deeper understanding of how different parameters impact model performance. I learned the significance of choosing appropriate activation functions, neuron counts, and learning rates, as well as how to handle data noise and adapt model architecture based on dataset complexity. A key challenge was finding the right balance between model complexity and generalization, particularly when dealing with complex datasets like XOR and spiral.

By experimenting with various parameters, I gained insights into the practical aspects of neural network design, including when to increase model capacity and how to effectively adjust hyperparameters. These lessons are crucial for building more effective neural networks that can be applied in real-world scenarios.

References

Dietterich, T. G. (1995). Overfitting and Underfitting in Machine Learning. *ACM Computing Surveys (CSUR)*.

Goodfellow, I., Bengio, Y., & Courville, A. (2016). *Deep Learning*. MIT Press. Retrieved from https://www.deeplearningbook.org/

Kingma, D. P., & Ba, J. L. (2014). Adam: A Method for Stochastic Optimization. *International Conference on Learning Representations (ICLR)*. Retrieved from https://arxiv.org/abs/1412.6980

LeCun, Y., Bengio, Y., & Hinton, G. (2015). Deep learning. *Nature*, 521(7553), 436-444.

Minsky, M., & Papert, S. (1969). *Perceptrons: An Introduction to Computational Geometry*. MIT Press.

Nair, V., & Hinton, G. E. (2010). Rectified Linear Units Improve Restricted Boltzmann Machines. *Proceedings of the 27th International Conference on Machine Learning (ICML)*. Retrieved from https://www.cs.toronto.edu/~fritz/absps/relu.pdf