TensorFlow Playground Exploration

#CYBERSEC520 #CLASS_4

Done by Gaurav Suryawanshi, Aditya Srikar Konduri, Mitali Joshi, Sejal Nayak

Learning Objectives:

Objective: Visual Exploration of Neural Network Parameters

Time Limit: 45 Minutes

Link: playground.tensorflow.org (note Regularization has been suppressed)

Setup: Work in pairs/teams of 2-3 students (in-person) or individually (online)

Pre-Exploration Setup

Everyone start here:

1. Go to playground.tensorflow.org
2. **Dataset**: Select "Circle" (default)
3. **Features**: Keep X_1 and X_2 selected (default)
4. **Hidden layers**: Start with 1 layer, 3 neurons
5. **Learning rate**: 0.03 (default)
6. **Activation**: ReLU (default)
7. **Click the "Play" button** to start training

Watch for 30 seconds - you should see the decision boundary evolving

---

Exploration 1: Activation Functions

Task A: Compare Activations

1. **Reset the playground** (refresh page)
2. **Setup**: Circle dataset, 1 hidden layer with 4 neurons
3. **Test each activation function and complete this table**:

Activation Function Comparison Table

Activation	Training Speed (fast/ medium/slow)	Final Test Loss	Converges? (Y/N)	Notes/Observations
ReLU	Fast	0.012	Yes	Smooth curve and fast Convergence
Tanh	Medium	0.012	Yes	Smooth Curve, not fast as ReLU

Sigmoid	Slow	0.013	No	Partial Convergence was Observed
Linear	Slow	0.507	No	No Convergence was Observed

What to observe:

- Which converges fastest? ReLU
- Which gets the lowest test loss? ReLu and Tannh (Didn't include sigmoid because partial convergence)
- Watch the loss curve - smooth or jumpy?

ReLU - > Smooth Tanh & Sigmoid - > Jumpy Linear - > No Convergence

---

Exploration 2: Universal Function Approximation

Task B: Can Neural Networks Learn Any Pattern?

Test the universal approximation theorem!

1. **Data Types**: Try each of these challenging patterns:
   - **Circle**: Can you separate with 1 layer? 2 layers?
   - **XOR**: This is the classic hard problem
   - **Spiral**: The ultimate test!
2. **For each data type, complete this table**:

Architecture Comparison Table

Dataset	1 Layer (6 neurons)	2 Layers (4,4)	3 Layers (4,3,2)	Top Architecture
Circle	Success? Yes Test loss: 0.0	Success? Yes Test loss: 0.0	Success? No Test loss: 0.524	1 Layer
XOR	Success? No Test loss: 0.520	Success? Yes Test loss: 0.336	Success? No Test loss: 0.569	2 Layer
Spiral	Success? No Test loss: 0.691	Success? No Test loss: 0.580	Success? No Test loss: 0.504	None

Tips:

- Let each run for at least 500 epochs
- "Success" = decision boundary cleanly separates the classes
- If stuck, try different learning rates

Task C: The Power of Depth

Focus on the XOR problem (the classic test):

Avg. Epochs: 900

XOR Challenge Results:

- 1 layer with 2 neurons: Success? No
- 1 layer with 4 neurons: Success? No
- 1 layer with 8 neurons: Success? No
- 1 layer with 16 neurons: Success? No
- 2 layers with 2 neurons each: Success? Yes
- 3 layers with 2 neurons each: Success? Yes
- Other (feel free to explore more on your own

3 Layes 4,2,2 Neurons Success? Yes

2 Layes 3 Neurons each Success? Yes

Exploration 3: Feature Engineering Effects

Task D: The Importance of Features

1. **Dataset**: Select "Circle"
2. **Features**: Start with just X_1 and X_2
3. **Network**: 1 layer, 4 neurons, ReLU
4. **Train it** - note the test loss

Feature Engineering Impact Table

Features Selected	Test Loss	Training Speed	Notes
X_1, X_2 only	0.475	0.3	Baseline
+ X_1^2 (x_1 squared)	0.385	0.1	
+ X_2^2 (x_2 squared)	0.286		
+ X_1X_2 (interaction)	0.236	0.1	
+ sin(X_1), sin(X_2)	0.201	0.1	
All features	0.252	0.1	

Exploration 4: Gradient Behavior

Task E: Learning Rate Experiments

Setup: Same spiral dataset, 2 hidden layers with 4 neurons each (4,4):

Learning Rate Effects Table

Learning Rate	Behavior Description	Converges? (Y/N)	Speed (fast/med/slow)	Final Test Loss
0.001	No Curve	No	Slow	0.426
0.01	Lot of Noise but convergence is observed	Yes	Slow	0.214
0.1	Bumpy Curve but Convergence is observed	Maybe?	Fast	0.355
1.0	Non-Linear Convergence is occasionally observed	Maybe?	Slow	0.416

Individual Reflection Questions

Answer these questions based on your exploration:

Question 1: Universal Approximation in Practice

You tested the universal approximation theorem with different datasets. When did deeper networks help more than wider networks? What does this suggest about real-world applications?

For complicated datasets such as Spiral and XOR, my group and I found that the deeper networks were better than the wider ones. These deeper architectures excel because they learn hierarchical feature representations which enable them to model complex and/or nested relations in the data. In real-world applications these are better equipped to understand difficult problems by building multiple abstract layers than expanding a sole, singular layer.

Question 2: Feature Engineering vs Architecture

Compare the impact of adding engineered features vs. making the network more complex. Which approach was more effective for the Circle dataset? Why might this matter in cybersecurity applications?

The feature engineering proved more useful in the circle dataset than just expanding the network depth and size. After creating features such as interaction and/or squared in terms of X1 and X2, the model provided helpful information. This enabled it to learn the pattern better and in an efficient manner. Conclusion for me was that this stresses upon the importance of feature engineering where in real-world scenario it can reduce computational expenses while enhancing the results and performance.

Question 3: Learning Rate Selection

Describe what you observed with different learning rates. How would you choose an appropriate learning rate for a real cybersecurity model?

Low learning rates such as 0.001 can have steady and slow training.

Moderately learning rates (0.01, 0.1) have a good balance so that training is fairly quick yet stable learning curve.

Very high rates (1.0) easily cause training instability or can't fully train, from my experience.

For a real-world use case in cybersecurity, I would choose moderately learning rates for getting a decent trade off between speed and stability, and watch the loss curve for monitoring adjustments, if the model's training becomes extremely slow.

Submission Instructions

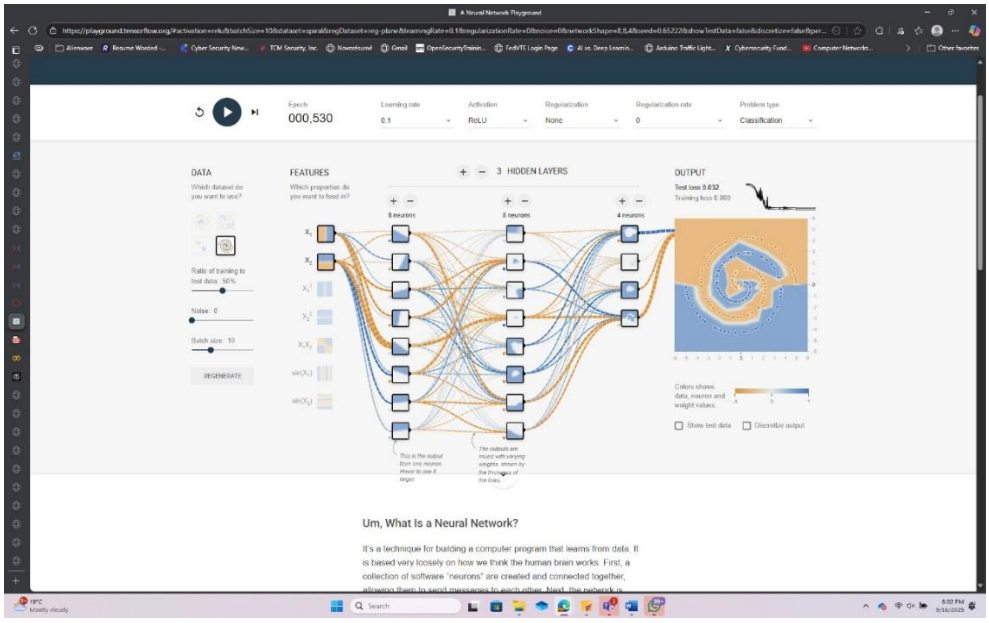
Submit via Canvas:

1. **All completed tables** (screenshots or typed)
2. **Reflection questions** (typed responses)
3. **Screenshot** of your most successful spiral dataset solution

---

Resources

- **TensorFlow Playground**: playground.tensorflow.org
- **Save configurations**: URL automatically updates with your settings
- **Help**: Hover over any element for explanatory tooltips

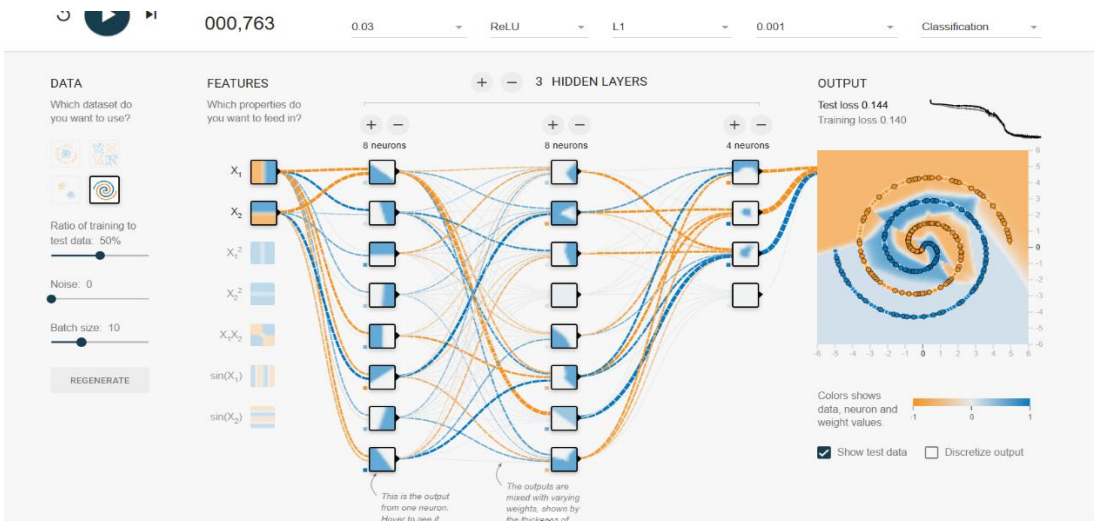


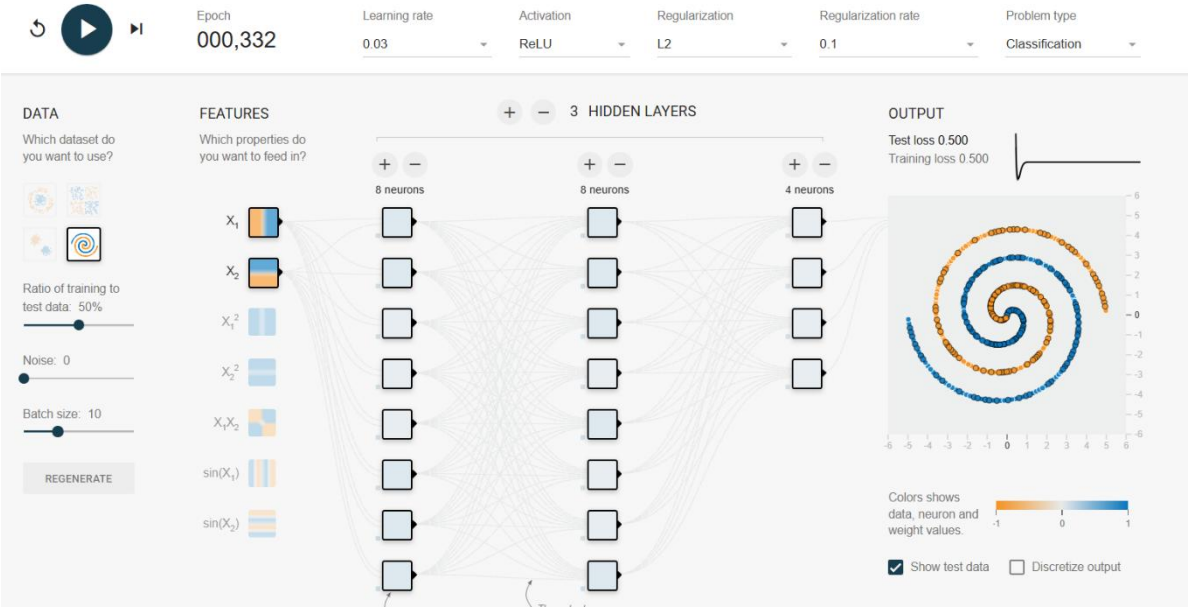
Bonus Challenges (Optional)

For fast teams/individuals:

1. **Regularization**: Try the L1 and L2 regularization sliders. What happens with high regularization?

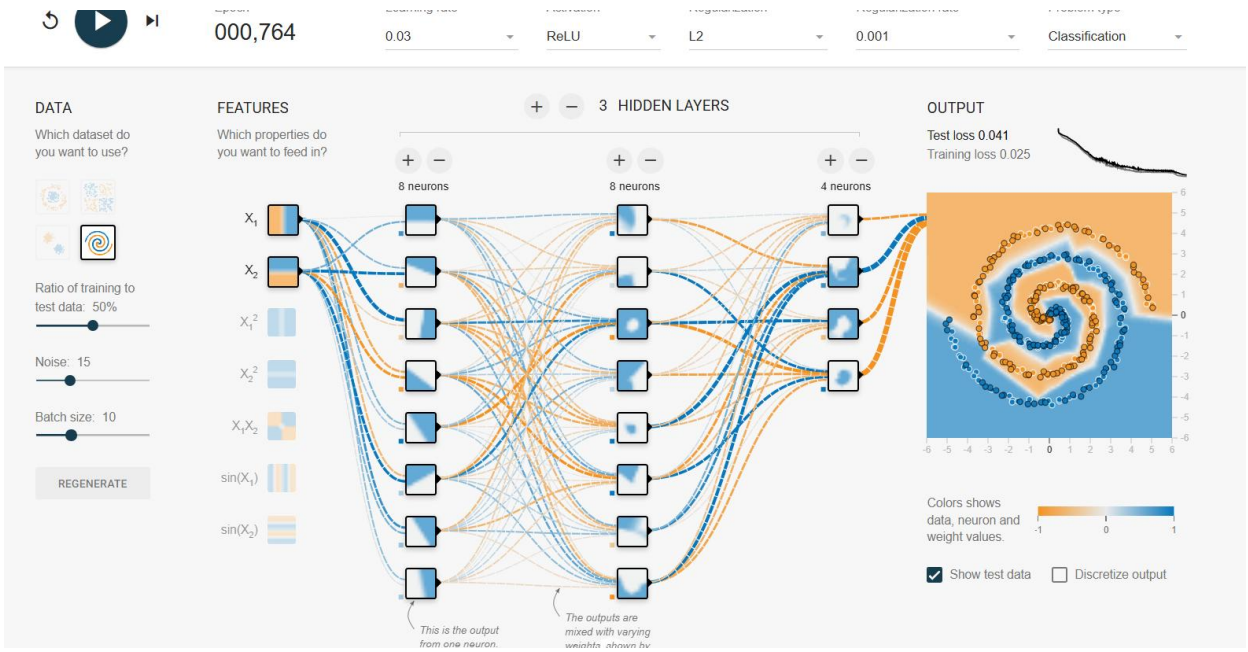
When I turned up L2 regularization, I saw that all the connections in the network got weaker, and the model drew a smoother, simpler boundary between the classes. At 0.001 and 0.01, this helped because the model stopped memorizing the data too closely and did better on new points, even if training accuracy dropped a little. But at 0.1, it became too simple, couldn't capture the patterns well, and both training and test accuracy went down. With L1 regularization, instead of just weakening connections, many of them disappeared completely, so the network used fewer paths to make decisions. At 0.001 and 0.01, this made the boundary simpler and still useful, but when I pushed it to 0.05, the model threw away too many connections, underfit the data, and accuracy dropped.





2. **Noise**: Add noise to your data. How does this affect training?

When I set L2 regularization to around 0.01 and start with no noise, the model trains smoothly, the decision boundary is clear, and both training and test accuracy look good. As I increase the noise to about 15% and reset, the data points begin to scatter, the boundary becomes less precise, and test accuracy drops. Pushing the noise further to 30–40% makes the classes blur together even more, the decision boundary simplifies a lot, and accuracy falls further. Noise essentially adds random jitter to the inputs, which makes it harder for the model to separate classes. This raises training loss and weakens generalization, but using a slightly higher L2 regularization (like 0.02–0.05) could help prevent the model from overfitting to the noisy points.


000,764
0.03
ReLU
L2
0.001
Classification
DATA
Which dataset do you want to use?
Ratio of training to test data: 50%
Noise: 15
Batch size: 10
REGENERATE
FEATURES
Which properties do you want to feed in?
3 HIDDEN LAYERS
8 neurons
8 neurons
4 neurons
OUTPUT
Test loss 0.041
Training loss 0.025
Colors shows data, neuron and weight values.
Show test data
Discretize output
This is the output from one neuron.
The outputs are mixed with varying