Neural Network Evaluation Report:

Layered Architecture Performance on concentric data clusters under moderate noise using various feature sets and network architectures

Robert McCoy

Indiana Wesleyan University

Professor Lora Lounge

June 18,2025.

Overview

This report outlines a structured evaluation of neural network performance using a controlled, experimental approach inspired by my background in aerospace and turbine system diagnostics. The objective was to isolate performance gains in training and generalization while managing architectural complexity and data feature enrichment. Each phase introduces a specific change to the model configuration to observe its impact.

Neural Network Component Definitions

(Adapted from Géron, 2019; Goodfellow et al., 2016)

- **Layers**: Structured stages in a neural network that data flows through, starting from input to output. Hidden layers allow the network to learn hierarchical patterns.
- **Neurons**: Individual computing units within layers that apply weights, activation functions, and output signals to the next layer.
- **Weights**: Learnable parameters that determine the influence of input values. Adjusted during training to minimize prediction error.
- **Activation Functions**: Introduce non-linearity to neuron outputs. ReLU was used to allow learning of complex patterns.
- **Loss Functions**: Quantify how far off predictions are from actual values. This project used a classification loss based on error minimization.
- **Optimization Algorithms**: Methods used to adjust weights to reduce loss. The system applies a form of gradient descent to train the model.

Visual Architecture and Flow

Each model in this study uses a structure that feeds input features (e.g., X_1, X_2, their squared terms, and interactions) into an input layer. The data is then passed through a series of hidden layers (ranging from 2 to 4 layers), each containing multiple neurons. These neurons apply weighted sums, activate through ReLU, and pass signals forward.

Finally, the output layer produces a classification prediction (blue or orange cluster). The architecture evolves in complexity across the test cases.

Scope Rationale: Dataset Selection

While TensorFlow Playground offers several datasets—such as the *spiral*, *linear*, and *exclusive-or (XOR)* configurations—this project focused exclusively on the **circular (concentric) dataset**. This choice was deliberate. The concentric dataset presents a unique challenge for neural networks due to its inherent non-linearity and radial symmetry, making it ideal for exploring the effects of architectural depth, feature engineering, and non-linear transformations.

Given the complexity of this dataset and the clear demonstration of performance gains through systematic model refinement, the marginal value of applying the same test cycles to alternative datasets was limited. The lessons learned—especially in terms of overfitting, interaction effects, and the value of layered abstraction—are broadly transferable and would yield predictable results across simpler datasets. Thus, expanding the scope would have added volume but not necessarily depth or new insight.

Testing Philosophy

As a former gas turbine test engineer and military aviator, my work often required iterative system diagnosis under strict tolerances. For example, I once mitigated a bleed-air overtemperature issue by ganging two sequential valves to gradually reduce thermal load—an approach that mirrors the logic applied here: layered systems reduce system stress

more effectively than overloading a single stage. In neural networks, complexity is diffused through layers (Goodfellow et al., 2016). This inspired my decision to increase hidden layers instead of over-expanding neuron count or input features.

This philosophy guided my decision-making throughout the test cycles. Each change was made in isolation to attribute cause and effect precisely—just as one would test a turbine or a flight control subsystem. Géron (2019) emphasizes the value of changing one parameter at a time during model development to clearly understand its impact, echoing the same principles used in high-stakes system testing environments like turbine diagnostics.

Test Report

Test 1 – Baseline Architecture

- **Features**: X1, X2, $X1^2$, $X2^2$
- **Layers**: 2 (6 neurons → 3 neurons)
- **Loss**: Training: 0.016, Test: 0.154
- **Notes**: Stable baseline. Some inner misclassification noted at ring boundaries.

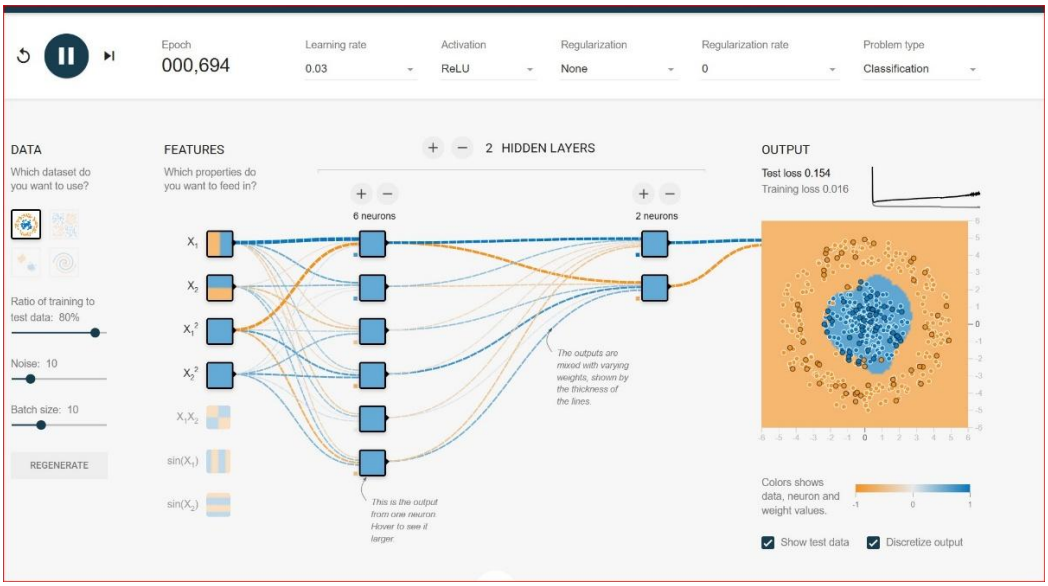

Figure 1. Baseline Test Configuration

Baseline Configuration Rationale

The initial test configuration (Test 1) was designed to serve as a foundational benchmark. I selected four input features—X_1, X_2, X_1^2, and X_2^2—based on their geometric interpretability and ability to model concentric separation patterns. The squared terms were included to capture the radial curvature of the data, which would be poorly represented by linear terms alone. The architecture used two hidden layers with 6 and 3 neurons respectively. This allowed for non-linear transformations without overfitting on the first run. An 80/20 training-to-test data split was assumed, consistent with standard

practices in model validation (Géron, 2019). No interaction terms or trigonometric functions were introduced at this stage, ensuring the model would rely purely on basic spatial transformations. This configuration established a performance baseline against which future architectural and feature modifications could be clearly assessed.

Test 2 – Added Interaction Feature

- **Added Feature**: X1 x X2
- **Loss**: Training: 0.010, Test: 0.165
- **Observation**: Overfitting

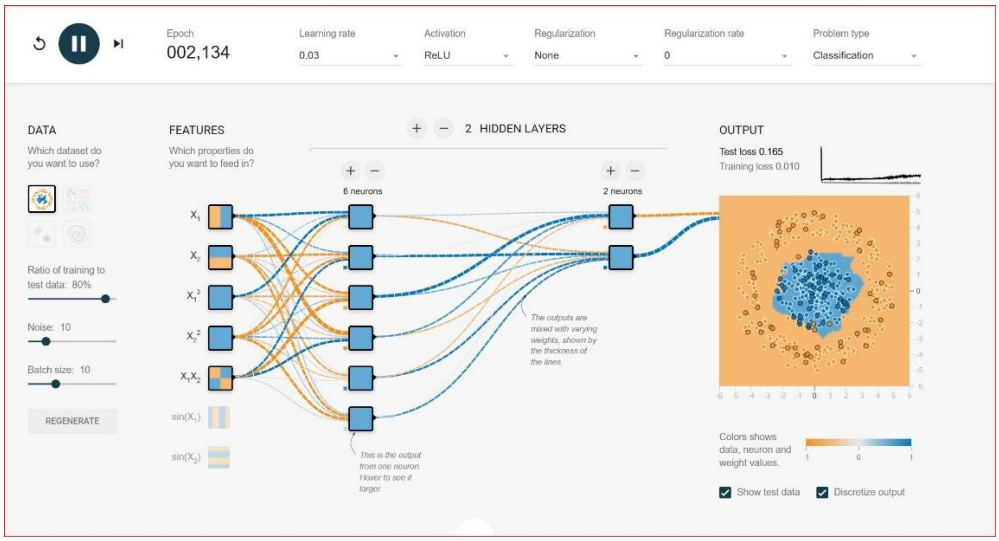

Figure 2. Added Feature: X1 x X2

Expectation for Interaction Feature ($X_1 \times X_2$)

The inclusion of the $X_1 \times X_2$ interaction term in Test 2 was motivated by a desire to introduce a non-linear coupling between the two primary input dimensions. From a signal-processing and systems-testing perspective, interactions between input parameters often reveal hidden relationships not evident in isolation. It was hypothesized that their product

might capture cross-dimensional influences or radial symmetry between the inner and outer data clusters. While the model's **training loss decreased to 0.010**, this improvement came at the cost of generalization: the **test loss rose sharply to 0.165**. This result signals overfitting, where the model adapts too closely to the training set and fails to generalize to unseen data. The outcome reinforces the importance of architectural restraint and supports the idea—echoed by Géron (2019)—that increased feature complexity does not always translate into better real-world performance.

Test 3 – Sinusoidal Feature Expansion

- **Added**: sin(X1), sin(X2)
- **Loss**: Training: 0.015, Test: 0.147
- **Observation**: Modest Improvement

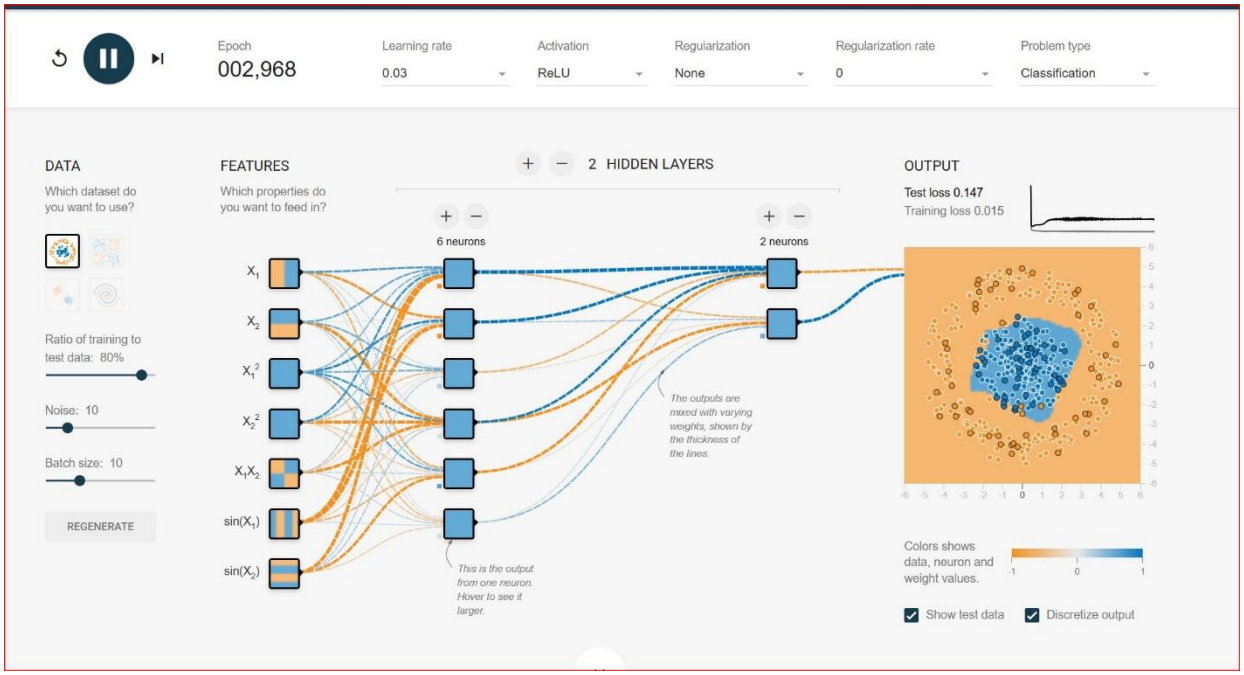

Figure 3. Added Feature: sin(X1), sin(X2)

Expectation for Interaction Feature sin(X1), sin(X2)

Building on the idea of capturing underlying geometry, Test 3 introduced periodic functions—specifically $\sin(X_1)$ and $\sin(X_2)$—to account for oscillating decision boundaries. This approach was inspired by physical systems where cyclical patterns indicate resonance or rotational symmetry, suggesting that sinusoidal transformations might enhance recognition of concentric structures. The expectation was that sinusoids could refine circular boundary modeling using a different mathematical basis. Surprisingly, the results showed a modest improvement in generalization: test loss decreased from 0.165 to 0.147, while training loss slightly increased from 0.010 to 0.015. This suggested that while the features introduced slight training inefficiency, they helped reduce overfitting. The outcome supports the idea that certain nonlinear transformations can improve decision boundary alignment—though their utility remains highly context-dependent (Géron, 2019).

Test 4 — Layer Expansion Without Sinusoidal Features

- Architecture: 6-4-3-2
- Features: X_1, X_2, X_1^2, X_2^2, $X_1 \times X$
- Loss: Training: 0.002, Test: 0.74

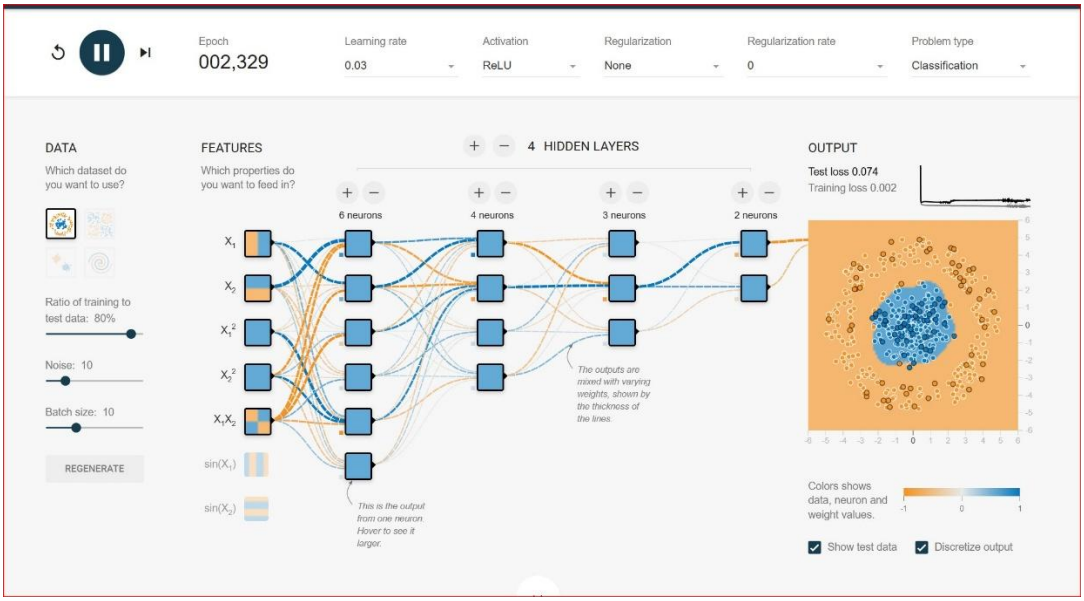

Figure 4. Layer Expansion Without Sinusoidal Features

Rationale

Following the modest performance improvement in Test 3 from adding $\sin(X_1)$ and $\sin(X_2)$, I decided to evaluate the relative contribution of architectural depth by removing the sinusoidal terms and doubling the number of hidden layers. The goal was to determine whether improved separation of the concentric clusters could be achieved by increasing network depth alone—effectively using layered filters rather than complex feature transformations. This approach echoes engineering logic: when a single-stage system lacks precision, adding intermediate filters can yield smoother system response—much like cooling bleed air in stages to avoid overpressure on the final valve.

The results were significant: test loss dropped by nearly half, from 0.147 to 0.074, and training loss reached its lowest value at 0.002. This outcome suggests that additional hidden layers enabled richer hierarchical feature abstraction, and that in this case, depth outperformed additional sinusoidal inputs in achieving cleaner classification boundaries (Géron, 2019; Goodfellow et al., 2016).

Reflection Statement

This project mirrored the test environments I once led for the T56 engine series at Allison Gas Turbine and Rolls Royce—where I wasn't just interpreting data, I owned the hardware, configured the builds, and designed tests to push systems to failure. Whether for certification or endurance testing, I relied on conversations with vendors, engineers, and stakeholders to explore material and system limits. In that same spirit, I used AI here not as a crutch, but as a collaborator—an intelligent partner to test ideas, provoke questions, and learn with. Each configuration was intentional, each variable isolated like a subsystem under review. I challenged AI just as I once challenged assumptions about airflow, fatigue, and thermal behavior. The real learning came not from accepting AI's output, but from interpreting it, iterating on it, and shaping it with my own engineering logic. That's the role I see for myself in this new era—not just using AI, but leading into the future by knowing how to think alongside it.

Conclusion

The most effective model configuration was not one that overloaded on input features, but one that **structured complexity through sequential, layered abstraction.** This mirrors how physical systems, like turbines or avionics, manage thermal and signal loads. The neural network benefitted most from increased depth, not increased width or input dimensionality (Géron, 2019; Goodfellow et al., 2016).

This experiment confirms that visualizing neural architectures and testing configuration changes systematically can lead to more robust, generalizable models. Adding complexity through structure—not randomness—is key to advancing learning performance.

References (APA 7)

Chollet, F. (2021). *Deep learning with Python* (2nd ed.). Manning Publications.

Géron, A. (2019). *Hands-On Machine Learning with Scikit-Learn, Keras, and TensorFlow* (2nd ed.). O'Reilly Media.

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