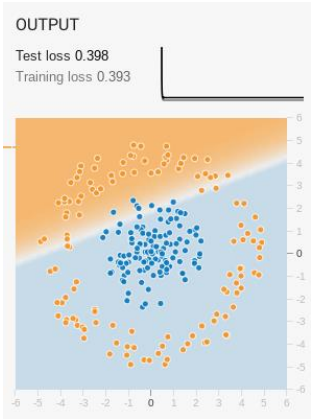
REPORT

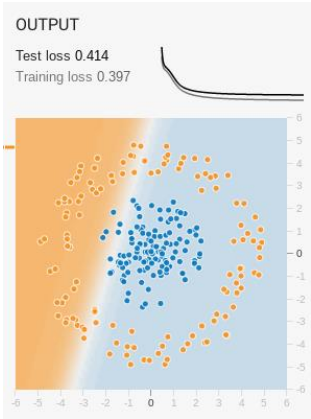
Introduction – A neural network is a machine model inspired by the anatomy of the human brain, designed to recognize patterns and learn from data. Their purpose is to recognize patterns in data and assist in classifying or predicting continuous values. In other words, neural networks are particularly resourceful for solving classification and regression problems. Neural networks are comprised of multiple key components: neurons (nodes), layers, and activation functions. Neurons are the building blocks of a neural network, they receive the inputs (also known as features), process it, and produce an output. Neurons are organized into layers, which allow the model to learn complex datasets. Multiple layers can exist in a neural network. Another component of neural networks are activation functions, they introduce nonlinearities and determine whether a neuron becomes activated or not. Activation functions also limit the range of values a neuron can have.

Task Overview - Each task delivered valuable insights that clearly demonstrate how different configurations, tools, and parameter adjustments impact the performance of the network.

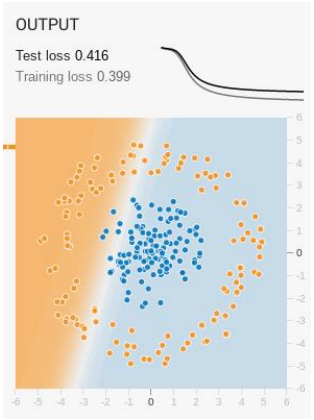
Task 1: Activation Functions - The first task dealt with activation functions. A neural network composed of one hidden layer was made and observations were made using multiple activation functions. Activation functions are mathematical functions that determine whether a neuron is activated or not. They take the input from the neuron and produce an output. The purpose of the activation function is to introduce non-linearity into the model, allowing the network to learn complex patterns and relationships in the data. Four activation functions were examined: ReLU, Tanh, Sigmoid, and Linear. Given our neural network comprising of one hidden layer. The model struggled solving the classification problem with all four activation functions.

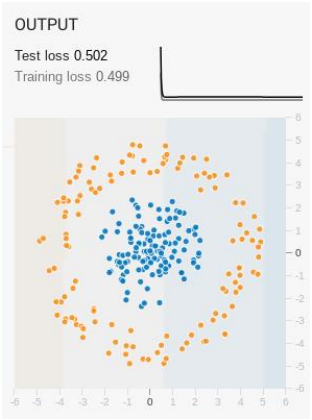


ReLU Output



Tanh Output





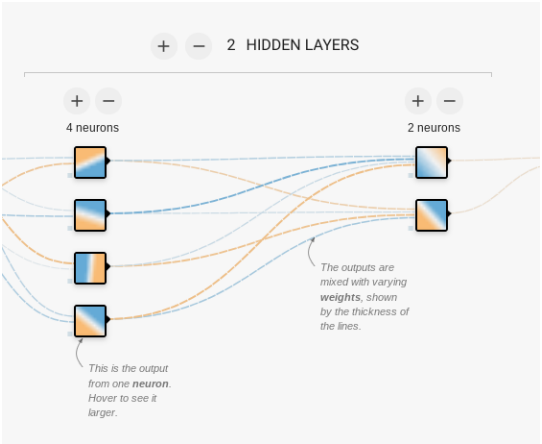
Sigmoid Output *Linear Output*

None of the four activation functions significantly influenced the output results in finding a solution. The test loss score plateaued around roughly 0.450 for all functions. One possible reason for this is the simplicity of the model used, which may have limited the effective utilization of the activation functions.

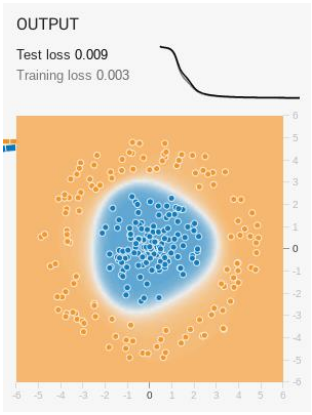
References: "Neural Networks Explained." *YouTube*, uploaded by [NeuralNine], 07 Dec. 2022, www.youtube.com/watch?v=rti0Ozfeqn8&t=425s.

"TensorFlow Playground – Neural Network Visualization Tool." *TensorFlow Playground*, https://playground.tensorflow.org.

Task 2: Hidden Layer Neurons – The number of neurons and the number of hidden layers were increased to see the effects of the network's performances. The Neuron Network composes of 2 hidden layers with 4 neurons in the first layer followed by 2 neurons in the second layer.

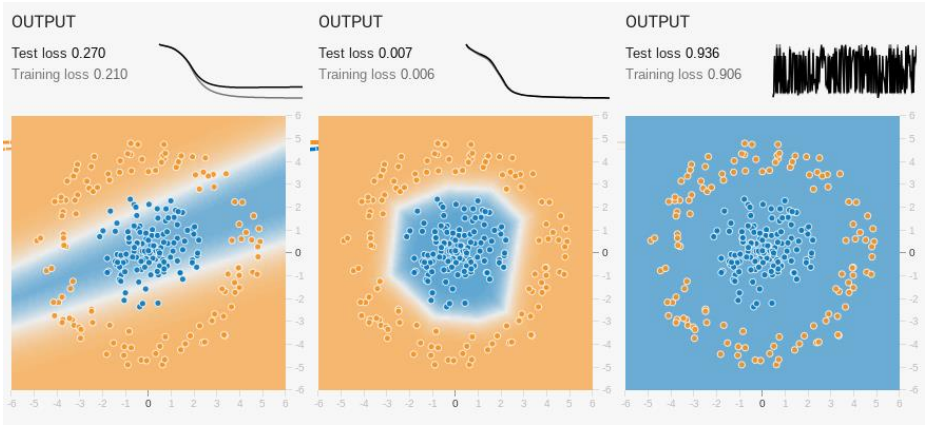


Neurons are the fundamental building blocks of a neural network. They receive inputs, also known as features, process them, and produce outputs. Organized into layers, neurons enable the model to learn from complex datasets, with multiple layers enhancing its capability to capture intricate patterns.



After running the model, ReLU, Tanh, and Sigmoid successfully produced outputs with test losses below 0.010. The primary difference among the three was the number of epochs needed to reach a solution. ReLU was the fastest, achieving convergence in 59 epochs, followed by Tanh at 80 epochs, while Sigmoid was significantly slower, requiring 700 epochs to reach the solution. One reason for the difference in speed is the behavior of the activation functions. ReLU maintains a constant gradient for positive inputs, contributing to faster convergence, while both Tanh and Sigmoid produce very small gradients, which can hinder learning. Linear had no success due to the problem type being classification.

Task 3: Learning Rate - The learning rate was adjusted to understand the impact on convergence speed and accuracy. The learning rate controls how much to change the model in response to the estimated error each time the model weights are updated. It plays an important role in optimizing the model and can impact the model's performance. Up to this point, the neural network model had a learning rate of 0.03. Observations were made with learning rates: 0.001, 1, and 3. The model still features 2 hidden layers with 4 neurons in the first layer and 2 neurons in the second layer. After testing the model with three different learning rates, the outputs varied significantly. The learning rate of 1 produced a result like 0.03, but more quickly. In contrast, the learning rate of 0.001 failed to converge, plateauing around 0.270. The learning rate of 3 was highly unstable, causing erratic fluctuations in the outputs.



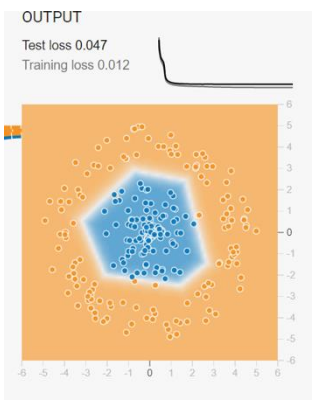
Learning Rate: 0.001 *Learning Rate: 1* *Learning Rate: 3*

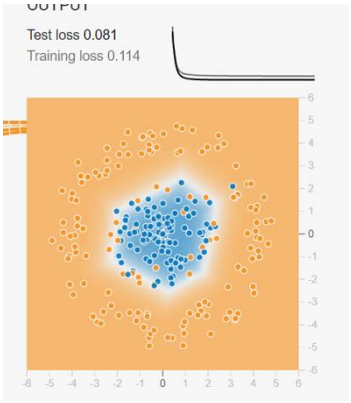
A learning rate of 0.03 likely provided a balanced step size allowing the model to update its weight efficiently without stagnating or becoming unstable.

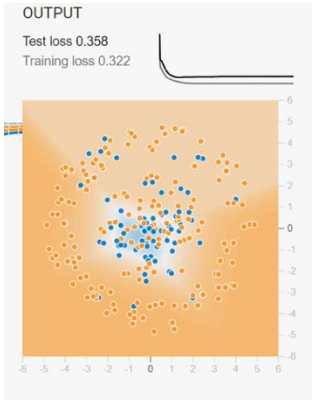
References: "Impact of Learning Rate on a Model." GeeksforGeeks, 6 Oct. 2020, www.geeksforgeeks.org/impact-of-learning-rate-on-a-model/.

Task 4: Data Noise - Through this experiment, I got a firsthand look at how noise in data directly affects a neural network's ability to generalize. Generalization refers to how well the model performs on new, unseen data, not just the training set. In an ideal scenario, the model should be able to apply the patterns it learned during training to make accurate predictions on data it has never encountered before. However, noise in the data can seriously mess with this ability. When I introduced noise into the dataset using TensorFlow Playground, it became clear how the network starts to struggle. With low levels of noise, the network was still able to learn the actual underlying patterns and generalize well to new data. But as I increased the noise, the network began to treat some of that noise as part of the pattern, which caused overfitting. Essentially, it was learning irrelevant details—random variations in the data that didn't represent anything meaningful.

At high noise levels, the generalization ability took a serious hit. The network created highly complex, erratic decision boundaries because it was trying to accommodate all the noise. As a result, its performance on new data plummeted because the model was no longer focusing on the real, important patterns—it was distracted by all the random noise. Data noise is basically any random or irrelevant information that gets mixed into the dataset. In real-world datasets, noise could come from faulty sensors, human error, or inconsistencies in how data is collected. The problem with noise is that it confuses the model during training because it doesn't represent the true patterns, we want the model to learn. In a neural network, the goal is to find patterns that help make accurate predictions. But when noise is present, the network may learn both the real patterns and the noise, treating it as if it's just as important. This leads to overfitting, where the model performs great on the training data (because it's memorizing everything, even the noise) but poorly on new data. The ability to generalize effectively is lost because the model can't distinguish between meaningful data and irrelevant noise.







***Noise:* 10** ***Noise*: 25** ***Noise:* 50**

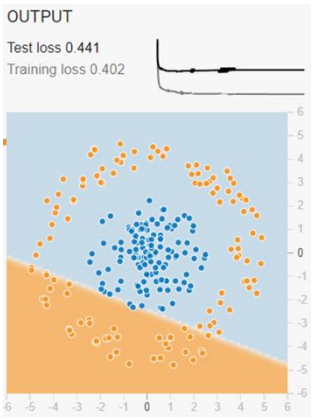
From my experiment, I realized how it became obvious that the more noise I added to the data, the harder it was for the network to generalize. At low noise levels, the network could find the true patterns and generalize well. However, as I increased the noise, overfitting became a big

issue. The model couldn't distinguish between meaningful data and irrelevant noise, and its performance on new data suffered as a result.

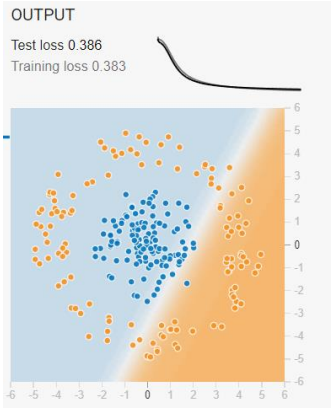
References: "Understanding Neural Networks." *Analytics Vidhya*, 10 Dec. 2022, www.analyticsvidhya.com/blog/2020/07/neural-networks/.

Task 5: Dataset Exploration—TensorFlow Playground is an interactive web-based platform where we can explore neural networks by training them with built-in datasets. There are four datasets: circle, exclusive OR (XOR), gaussian, and spiral.

Using the default features X1 and X2, with one hidden layer and 1 neuron. When we first tested the circle data with 1 hidden layer and one neuron, we lost more than 0.4 test and training losses in 100 epochs. It's clear that without more neurons and an extra hidden layer, it will not perform as well as it could. The best activation without any modification is Tanh; at 000,184 epoch, it reached the 0.386 test/training loss.

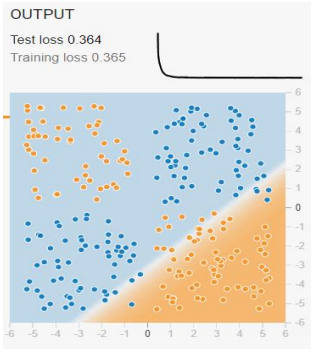


ReLU (Circle)



Tanh (Circle)

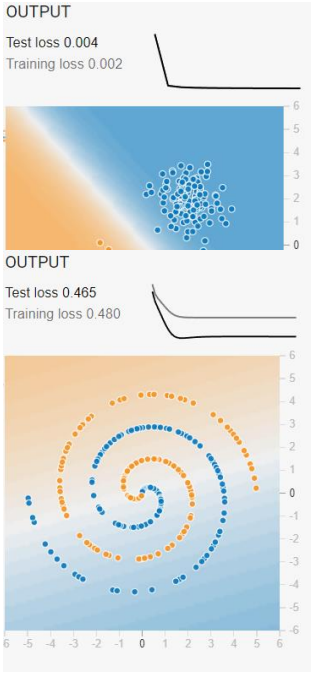
Moving on to Exclusive OR (XOR) with the same layout as the circle dataset, with the ReLu activation this dataset performed better with a test/training loss of 0.364 at 001,049 epoch without any hidden layers or input involved. In fact, ReLu was the best performing activation with Sigmoid coming in second at a training loss of 0.367.



Exclusive OR(XOR)

Now with the Gaussian dataset, something occurs that is not present in the other datasets. With bare modifications, this dataset is able to reach a staggering 0.004 test/training loss in just 000,013 epoch. The best activation is linear. I think this is because the negative and positive dots

are separated so it can be easily generated; also, adding any more hidden layers or inputs is not a good idea; it can unnaturally burn the system and produce little benefit.



Gaussian

Lastly, the spiral dataset, same as the other circle dataset, without any adjustment to the model and leaving it as is, the best performance this model can do is reach the lowest test loss of 0.465. When comparing the ReLU with the other activations, Linear is the best. This might not be the case if different features, input, and hidden layers are added.

Spiral

Selecting the right data set is crucial for efficiency and efficiency, saving you time and resources. When choosing your dataset take into consideration:

- Make sure you have enough data; it can affect your model performance as it may miss some patterns in the data. You can use techniques such as data augmentation or transfer learning to make up.
- All data should have the same samples as the other, this can lead to biased predictions. You can use techniques such as oversampling, under sampling, or class weighting.
- Also, all data points should not be too far from the other data samples, which can impact on your model's performance.

Understanding these parameters can ensure you use your time efficiently and are ready to select your dataset selection for neural network training.

Reference: https://medium.datadriveninvestor.com/deep-learning-with-tensorflow-playground-e6b194ee8fac

https://medium.com/@jan_marcel_kezmann/the-dos-and-donts-of-dataset-selection-for-machine-learning-you-have-to-be-aware-of-8b14513d94a

The Effects of Parameter Changes - The components of the neural network are critical in the search of a result we are looking for. Tweaking each component independently has influenced the output in a positive or negative direction. A reflect on our observations, we observed a relationship between the structure of our data model and the dataset. The more complex the dataset is, the more complex the model will need to effectively identify patterns and derive solutions. Neurons are the fundamental units that process input data; having more neurons in a model allows it to capture and analyze a greater amount of information. Additionally,

incorporating more layers of neurons adds the complexity needed to effectively handle intricate datasets. We observed the effects of activation functions, these functions determine which neurons will be activated through a mathematical process. Of the 4 observed, after adjusting neurons and layers, ReLU, Sigmoid, and Tanh were successful in deriving a solution for the classification problem. Differences in speed measurements were noted as ReLU took the least number of epochs to derive a solution. Learning rates determine how much to update the model's weights in response to the estimated error each time the model weights are updated. We observed how changing the learning rates drastically affect the performance of the model. The learning rate has a high sensitivity level. This means that small changes in the learning rate can significantly impact the training process and the final performance of the model. The balanced learning rate observed was 0.03. Observations of noise adjustments were observed, unwanted data affected the capabilities of the model in detecting patterns. A strong correlation between the level of noise and performance showed that the more noise there was in the data, the weaker the performance of the model. Lastly, observations were made with changing the datasets. The model performed the best using the Gaussian dataset. The Spiral dataset proved to be the most challenging for our model.

Practical Implications - Reflecting on my findings, we've come to realize just how important it is to understand how neural networks respond to parameter changes, especially in real world applications. In simple terms, data is often noisy, incomplete, or messy, and knowing how to adjust for that is crucial. What stood out the most was that during the TensorFlow Playground experiment was how quickly a model can overfit when there's noise in the data. This happens when the network starts learning from irrelevant information instead of focusing on actual patterns. In real-world scenarios- like predicting house prices or diagnosis diseases-this could models that perform well in training but fail when faced with unseen data. By weaking parameters like noise levels and model complexity, we now better understand how these adjustments can help a model generalize and perform reliably in less than perfect conditions.

This deeper understanding of neural network parameters has shown that tuning is essential for creating effective models. It's not enough to aim for the highest accuracy on training data; it has to ensure that the model can handle noisy, real-world data without losing its ability to generalize. Whether it's through regularization, adjusting network complexity, or managing noise, these techniques are practical tools that can be applied to ensure models don't just work well in theory but also can perform in real world situations. Lastly, learning to navigate these parameters is key to building neural networks that are robust, reliable, and able to deliver consistent results when it really matters.

References

"Introduction to Convolutional Neural Networks." *GeeksforGeeks*, 2 Aug. 2023, www.geeksforgeeks.org/introduction-convolution-neural-network/.

"Understanding Neural Networks." *Analytics Vidhya*, 10 Dec. 2022, www.analyticsvidhya.com/blog/2020/07/neural-networks/.

"Introduction to Convolutional Neural Networks." GeeksforGeeks, 16 May 2023, www.geeksforgeeks.org/introduction-convolution-neural-network/.

"Impact of Learning Rate on a Model." GeeksforGeeks, 6 Oct. 2020, www.geeksforgeeks.org/impact-of-learning-rate-on-a-model/.

"TensorFlow Playground – Neural Network Visualization Tool." *TensorFlow Playground*, https://playground.tensorflow.org.

"Neural Networks Explained." *YouTube*, uploaded by [NeuralNine], 07 Dec. 2022, www.youtube.com/watch?v=rti0Ozfeqn8&t=425s.

Data Driven Investor. (2019). *Deep learning with TensorFlow playground*. Retrieved from https://medium.datadriveninvestor.com/deep-learning-with-tensorflow-playground-e6b194ee8fac

Kezmann, J. M. (2020). *The dos and don'ts of dataset selection for machine learning: You have to be aware of*. Retrieved from https://medium.com/@jan_marcel_kezmann/the-dos-and-donts-of-dataset-selection-for-machine-learning-you-have-to-be-aware-of-8b14513d94a

Reflections - Alejandro

This week's group assignment involved becoming familiar with neural networks. We gained information about what neural networks are, what their functions are, the many components essential to it and the influences it has on the model's performance. The purpose of this reflection is to reinforce and help retain the important information I learned during the assignment.

We used a website that allowed us to play around with the model without worrying about computer limits and coding to gain a better understanding of neural networks. The website provides thorough explanations that I have found very useful. A detail explained in the playground was the use of colors. The colors orange and blue were used to visualize datapoint values. Data points that were orange meant that they were assigned a negative value and blue data points were positively valued. To get familiar with the terminology on the website, I used the PowerPoint in this week's module and very helpful YouTube videos.

If I had to pinpoint an area of the assignment I was fixated on, it would be the output. The output or the result is where I witness how each component functions after gathering information on them. Whenever the output stagnated for certain activation functions, I would seek out why. This is the part of the assignment where I found myself enjoying the assignment because it shows me that I was actively engaged in it and applying myself to learn more!

I have found this assignment to be very stimulating. I always start any assignment with a degree of discomfort because I worry about how I will be able to learn new material and effectively apply it to an assignment. Those feelings gradually diminished because I realized how the assignment was structured. The components that make up the neural network are separated into 5 tasks that help understand the changes they make to the model's performance. Essentially, the assignment is to understand! I believed after I had completed this assignment, I had gained valuable insights into neural networks, including their components and their respective purposes.

A lesson or sentiment I feel after every assignment – no matter which class it is – is that I must be more receptive. I have to be more receptive to new ideas and feedback. Each task challenges me to approach my work with an open mind, embracing the opportunity to learn from different perspectives. This mindset not only enhances my understanding of the subject matter but also allows me to have personal growth and become resilient. I realize that being receptive allows me to adapt and improve, ultimately leading to a deeper appreciation for the learning process.

Reflections-Jesus

This week's group assignment focused on exploring neural networks. I learned about their structure, functionality, and the key components that significantly impact their performance. This reflection aims to reinforce the essential concepts I grasped during the assignment.

We utilized an online platform that enabled us to experiment with neural networks with the constraints of coding or hardware limitations. The website offered detailed explanations that were incredibly helpful. One aspect that stood out to me was how it used colors to represent data point values. Orange indicated negative values, while blue signified positive ones. To familiarize myself with the technology, I referred to the Powerpoint slides from this week's module and watched informative Youtube videos.

What really got me into the assignment was analyzing the output generated by the model. I was fascinated by how each component influenced the results. Whenever the output stagnated with certain activation functions, I was curious to figure out why. This aspect of the assignment was particularly rewarding, as it fueled my curiosity and deepened my understanding of the material.

Initially, I approached this assignment with some anxiety about my ability to grasp the new concepts effectively. However, as I progressed, I appreciated the well-structured nature of the tasks. The assignment was broken down into five manageable sections, each shedding light on how different adjustments affected model performance. I emerged from this experience with a clearer understanding of neural networks and their various components.

One of the most significant lessons I learned from this assignment is the importance of embracing challenges and staying open-minded. Each task encouraged me to adopt a flexible mindset, allowing me to learn from different approaches and viewpoints. This adaptability not only enriched my understanding of neural networks but also contributed to my personal growth. I now recognize that maintaining an open attitude is crucial for navigating the complexities of learning, overall enhancing my overall experience in this course.

Reflection – Mayela

On this assignment we learned about neural networks and their behaviors. My task was to explore different datasets available in the TensorFlow playground and analyze how the network performs on each. This task was initially challenging because I was supposed to re cap without interrupting with the task of others.

This was initially challenging because i didn't know if you could modify the features, or add any hidden layers or input. At the end i just decided that the best way to explore all the data sets and see how they perform in the network is settle all datasets to default and measure the epoch time and test/training loss.

The simpler datasets like Gaussian benefit from minimal architecture, while complex datasets like Spiral and Circle need deeper networks. In this case, adding extra layers or neurons will not improve performance significantly but could introduce unnecessary computational costs, risk of overfitting, and wasted resources.

The choice of activation function plays a significant role in performance, with Tanh excelling in non-linear datasets and ReLU showing strength in XOR-type problems. Also the structure of the dataset determines the best network setup, making dataset understanding crucial for optimizing neural network performance. Overcomplicating or under-engineering a network can lead to inefficiencies or underperformance.

As seen with datasets like XOR and Spiral, feature interactions are essential. Hidden layers in neural networks act as feature generators, combining raw input features like X1 and X2 in TensorFlow in ways that capture more sophisticated patterns. When you're dealing with complex datasets, hidden layers help the network extract these non-obvious relationships that can't be captured by the input features alone. More hidden layers allow the network to model increasingly complex functions, but again, adding too many layers without proper regularization can lead to overfitting.

Reflection – Eduardo

Working with neural networks in TensorFlow Playground has been a really eye-opening experience for me. I thought I had a decent grasp of how these models work from reading and lectures, but actually getting hands-on with the parameters made everything click. Seeing how tweaking things like noise levels or the network structure could totally change the model's behavior really hit home. Neural networks are powerful, but they're also super sensitive to the tiniest changes. It was fascinating to watch how quickly they could go from nailing predictions to completely missing the mark just because of a little noise in the data.

One of the biggest takeaways for us was understanding how easy it is to overfit when there's noise involved. I had heard about overfitting before, but it didn't fully sink in until I watched it happen in real-time. As we cranked up the noise, the network started treating random junk as if it were important, which led to some pretty wild decision boundaries. The model was basically memorizing the noise instead of focusing on the real patterns. This is a big deal because, in the real world, data is often messy, and models need to learn how to ignore that noise if they're going to make reliable predictions.

We definitely faced a few challenges while doing this. At first, it was tough to figure out how to keep the model from overfitting when the noise levels got too high. We had to mess around with things like regularization and adjust the network architecture to get it back on track. It was a lot of trial and error, but eventually, we found a balance that worked. It taught us that building a good neural network isn't just about choosing the right algorithm—it's about fine-tuning and adjusting parameters until everything fits together.

Overall, this experience has given me a much deeper understanding of how neural networks behave in different conditions. It's one thing to read about it, but actually experimenting with these changes gave me a real sense of how sensitive these models are. Plus, I feel more confident now in my ability to troubleshoot when things don't go as expected. There's definitely a lot of trial and error involved, but that's part of the learning process 💻.