**Tensorflow playground**

Neural networks are subsets of machine learning and are inspired by the functions and structure of the human brain. Neural networks consist of interconnected processing units called neurons and are organized in layers that help the computer to recognize patterns and make decisions. Neural networks are very important in image recognition. Neurons are the basic unit of the neural network. They receive the input, apply weights and biases, and then process the sum through an activation function and pass the output to the next layer. The layers are what neural networks are composed of. The input layer receives the raw data, the hidden layers perform computations and extract features while more layers create a DNN (deep neural network). The output layer produces the final prediction or classification.

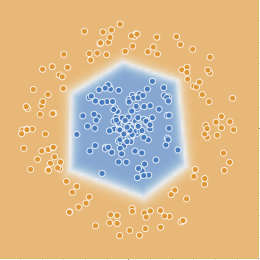
Activation functions are responsible for transforming the weighted sum of inputs into the output that determines whether a neuron should be activated. Each method has unique characteristics that make it suitable for specific tasks and network architectures. ReLU is a popular activation function in deep learning. It adds non-linearity, but stays fast. ReLU is easy to compute and helps models learn faster. It avoids vanishing gradients for positive inputs, too. But, ReLU has a problem: the dying ReLU issue where neurons can get stuck outputting zero and stop learning. Leaky ReLU and PReLU fix this by allowing small, negative outputs instead of just zero.

The Tanh activation is better than sigmoid. Tanh outputs values from -1 to 1. This helps balance values in hidden layers. It makes weight updates work better. But Tanh has issues like sigmoids. It struggles with the vanishing gradient problem. Big or small values make gradients get small. This slows down learning in deep networks.

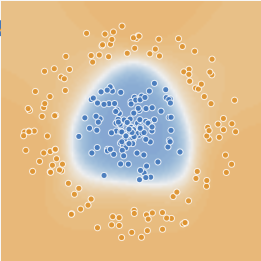
Sigmoid is great for binary classification tasks. It turns numbers into a range between 0 and 1. This makes it handy for guessing probabilities. But, sigmoid has some downsides. It can suffer from vanishing gradients. This slows down the learning process. Also, its outputs aren't centered around zero. This can make weight updates less effective. Even with these issues, sigmoid is still used. It often appears in the final layers of binary models.

Linear activation doesn't add a curve to the model. It's mainly for regression since the result needs to be a number. If you use linear activation in the middle layers, the network can't learn hard stuff. It will act like a basic straight-line model.

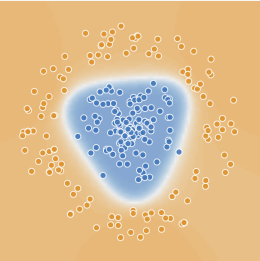
Changing the activation function to ReLU produces positive values or zero causing outputs to sparse because the negative inputs become zero. It speeds up the training which helps with deep networks but neurons can die meaning they become permanently inactive making them only output to zero and stop learning. This usually forms piecewise linear boundaries and it usually works well for complex data but can become an issue if too many neurons become inactive.



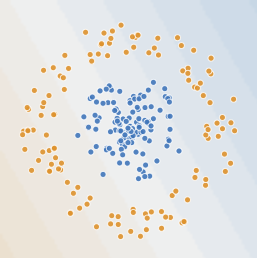
Changing the activation function to Sigmoid makes the output values range between zero and one. This impact makes the training suffer from the vanishing gradient problem. This is where for very large or small, the gradient approaches zero which slows down the learning process. The decision boundary for this activation function creates smooth and continuous decision boundaries but may struggle with sharp transitions leading to slower convergence.



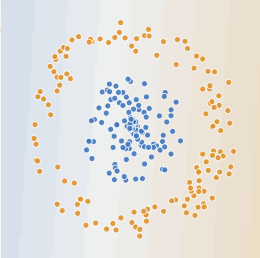
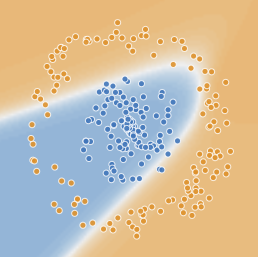
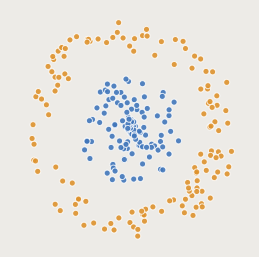
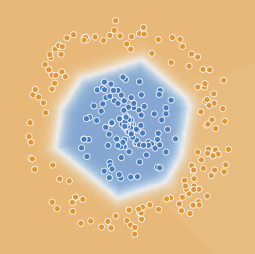
The Tanh activation function outputs from -1 to 1 making it zero centered which is what makes it different from sigmoid. This helps in balancing positive and negative activation which reduces bias in learning. Tanh also suffers from vanishing gradients, but not as much as Sigmoid. This encourages stronger activations because it maps input to a broader range. This works better in hidden layers compared to Sigmoid. Tanh is also more flexible than Sigmoid, leading to smoother and more effective separation of classes.



The output for linear activation functions are the same as the input. The network can only learn different linear relationships making it useless for complex patterns. It doesn’t have a vanishing gradient problem, but lacks the ability to model non-linearity. The decision boundary is always a straight line, making it ineffective for most real-world classification tasks and is used primarily in the output layer of regression models.

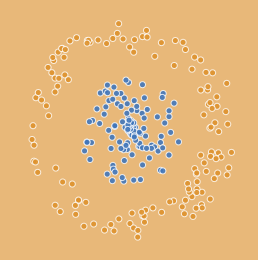
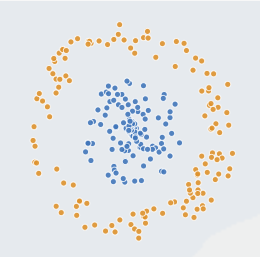
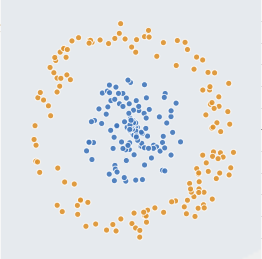


Adding more neurons allows the network to be able to learn more complex relationships but it also increases the chances of overfitting.The network can also underfit if the neurons are too low and then it won’t be able to capture the necessary data patterns. Adding more hidden layers can help the network learn increasingly complex data features and each layer can represent a higher level abstraction of the input. Adding too many layers can also result in overfitting if there is insufficient training data and the model can become harder to train because of problems like vanishing gradients. Below are images of what happens by increasing the hidden layers to 3 and increasing the neurons to 4, 3, and 2.



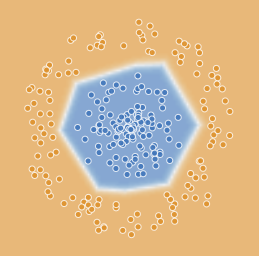
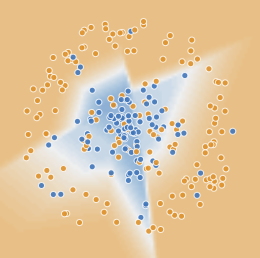
ReLU Sigmoid Tanh Linear

The learning rate controls how much the model's weights are adjusted during training in response to the estimated error and determines how fast or slow the neural network learns. A high learning rate allows the model to learn quickly but may cause instability which prevents convergence. A lower learning rate has a more precise weight update but can lead to slow convergence which requires more training iterations. I noticed when I put the learning rate at 10, the output is significantly different than if I put it at 0.00001. This is an example with the activation function on ReLU:



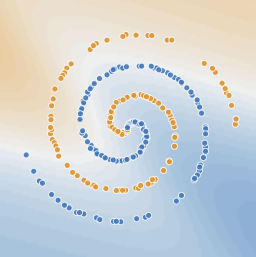
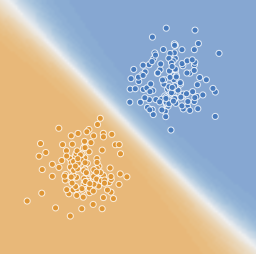
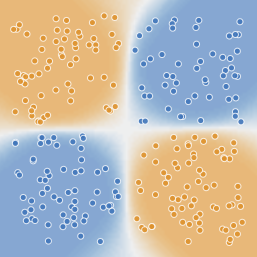
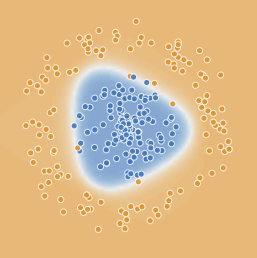
Start Output: 0.0001 rate Output: 10 rate

Data noise is the random variations or distortions in the input data that don’t represent true patterns but with inconsistencies and errors instead. Adjusting the noise to a low noise (0-10%) lets the patterns be more clean and defined and allows the decision boundary to be smooth and follow the data distribution. Moderate noise (10-30%) can learn patterns but starts to fit some noise and the decision boundary becomes slightly irregular with signs of overfitting. Generalization slightly decreases but the model can still make decent predictions. High noise ( anything above 30%) has large amounts of distortion which makes it a lot harder for the model to distinguish between pattern and random noise. Decision boundary is very distorted and has poor generalization in new data. Here are some examples with activation in ReLU:



Noise level: 50 Noise level: 5

The spiral dataset creates a complex and non-linear pattern and requires a deep network with multiple hidden layers to work effectively. Although more layers and neurons are needed to make it perform correctly, it is still susceptible to overfitting and it also takes longer to train than simpler datasets. The circle dataset is two classes, one forming an inner circle and another forming an outer ring. This dataset requires a nonlinear decision boundary for proper classification. The exclusive OR dataset are grouped in 4 sections where the diagonal opposites have the same label. A small network with 2 hidden layers can easily learn the pattern. Adding too many layers can lead to overfitting. The Gaussian dataset has some noise which makes classification harder. It has two overlapping Gaussian distributions that have different classes. Neural networks with a few hidden layers improve accuracy by handling slightly nonlinear patterns.



Circle Exclusive OR Gaussian Spiral

Understanding neural network parameters is crucial for developing effective AI solutions across real-world applications. The number of hidden layers and neurons plays a vital role in tasks like medical imaging, where deep networks capture intricate patterns, while in fraud detection, selecting the right dataset ensures accurate anomaly detection. Adjusting the learning rate is essential in fields like autonomous driving, where smooth adaptation to changing conditions prevents instability. Managing data noise is critical in speech recognition, ensuring voice assistants perform well in noisy environments. Optimizing these parameters balances accuracy, computational efficiency, and generalization, preventing issues like overfitting in stock market predictions or underfitting in medical diagnosis AI. Additionally, real-time applications, such as recommendation systems and chatbots, benefit from dynamic parameter adjustments to enhance adaptability. By fine-tuning these factors, AI models can perform reliably, efficiently, and accurately in complex, real-world scenarios.

Hands-on work with TensorFlow Playground gave me a better grasp of neural network learning. I saw how they change based on different settings. I changed hidden layers, neurons, learning rates, and noise and these experiments showed how each part affects a model’s ability to work well in new situations. I learned that hard datasets, such as the spiral, need deep networks. They also need special activation functions that bend the data. Simple datasets can use fewer layers to get the job done. Changing the learning rate showed how to balance speed and keep things stable. It is important to tune these settings to get the best results. One issue was balancing underfitting and overfitting. This happened when I changed the number of neurons and layers. If the model was too simple, it could not learn patterns. If it was too complex, it memorized the training data. Then, it could not work well with new data and to fix this, I tried different designs. I used techniques to prevent overfitting. I changed settings bit by bit. Another problem was noise in the data. This sometimes made the decision lines jump around. I found that using a fair number of neurons helped. So did using techniques to prevent overfitting. Overall, this work showed how important it is to know how neural networks act. Hyperparameter tuning affects how well they work. I learned how to design good models that work in the real world. By picking network designs carefully and tuning settings, we can make AI systems. These systems can be strong and steady.