

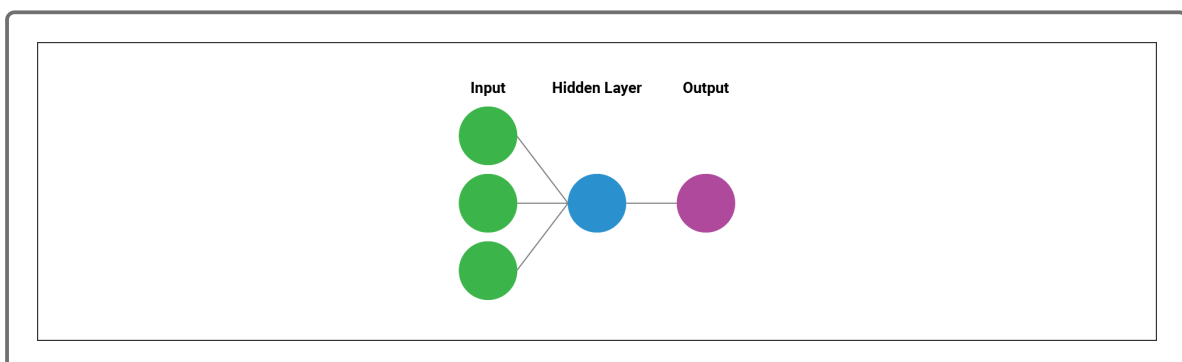
19.1.3

Make the Connections and Explore TensorFlow Playground



Beks has a solid understanding of the structure of a single neuron—now it's time to unpack the structure of the network.

Now that we understand the structure of a single neuron, we can begin to build the structure of a neural network:



A basic neural network has three layers:

- An **input layer** of input values transformed by weight coefficients
- A **single "hidden" layer** of neurons (single neuron or multiple neurons)
- An **output layer** that reports the classification or regression model value

As mentioned previously, neural networks work by linking together neurons and producing a clear quantitative output. But if each neuron has its own output, how does the neural network combine each output into a single classifier or regression model? The answer is an **activation function**.

The **activation function** is a mathematical function applied to the end of each "neuron" (or each individual perceptron model) that transforms the output to a quantitative value. This quantitative output is used as an input value for other layers in the neural network model. There are a wide variety of activation functions that can be used for many specific purposes; however, most neural networks will use one of the following activation functions:

1. The **linear function** returns the sum of our weighted inputs without transformation.
2. The **sigmoid function** is identified by a characteristic S curve. It transforms the output to a range between 0 and 1.
3. The **tanh function** is also identified by a characteristic S curve; however, it transforms the output to a range between -1 and 1.
4. The **Rectified Linear Unit (ReLU) function** returns a value from 0 to infinity, so any negative input through the activation function is 0. It is the most used activation function in neural networks due to its simplifying output, but it might not be appropriate for simpler models.

5. The **Leaky ReLU function** is a "leaky" alternative to the ReLU function, whereby negative input values will return very small negative values.

Match the activation function names to the correct image using the descriptions in this section for guidance.

Image A

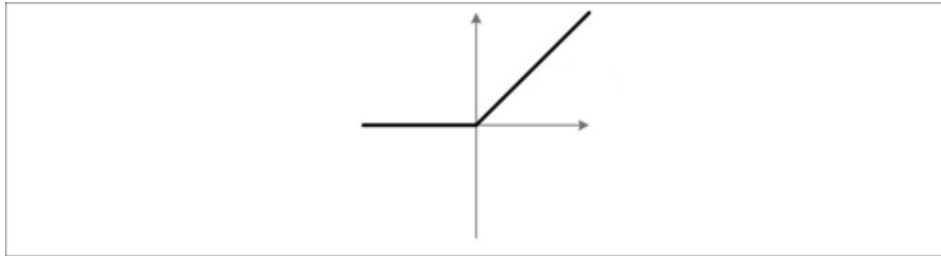


Image B

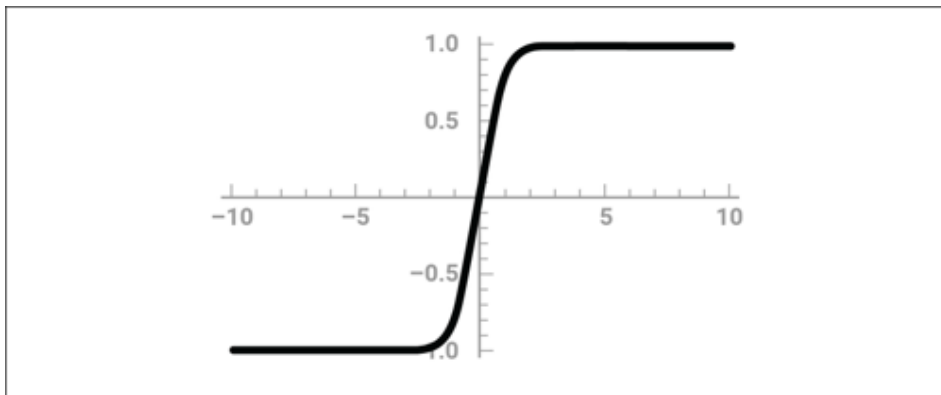


Image C

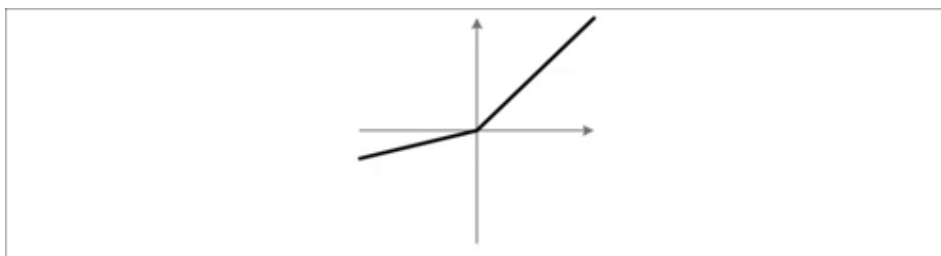
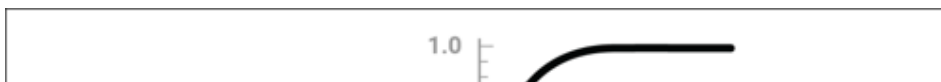


Image D



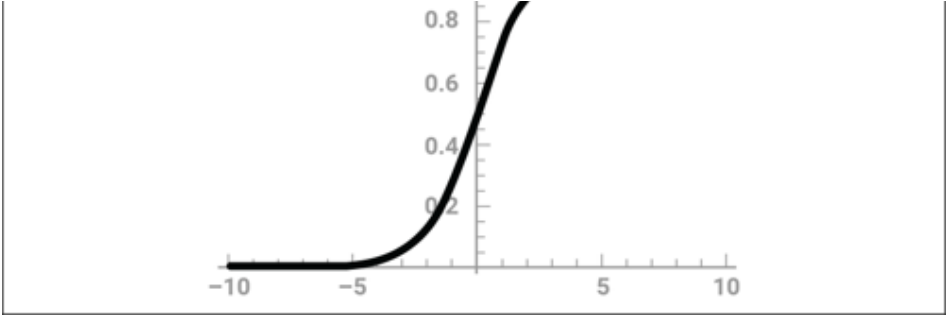


Image A:

Image B:

Image C:

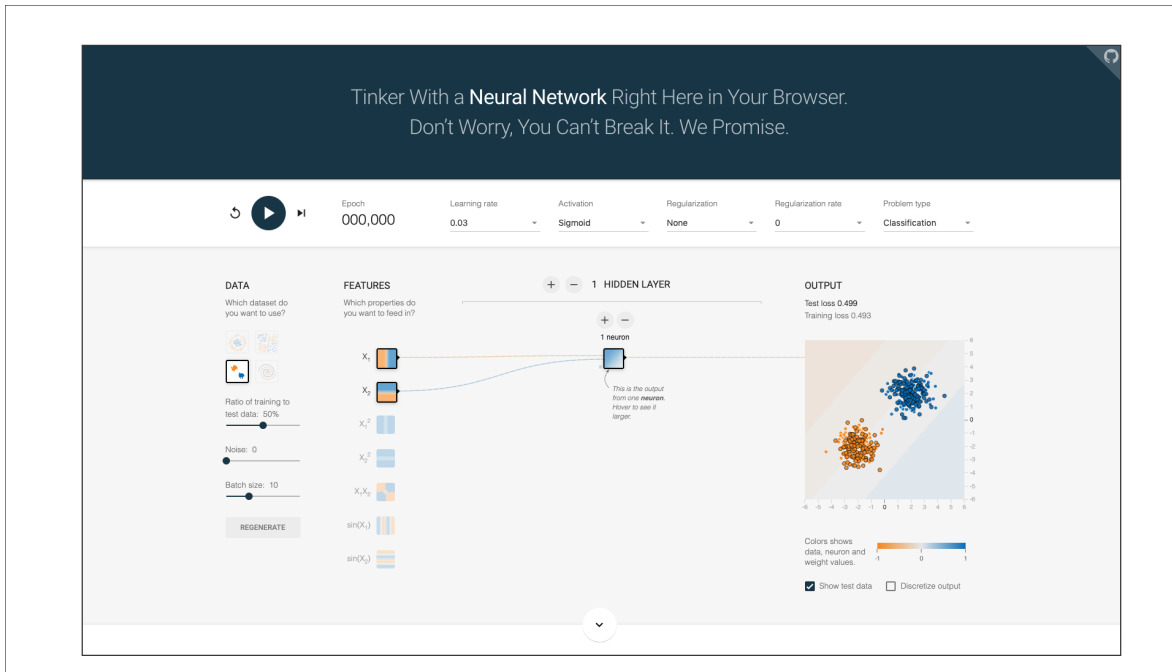
Image D:

To better understand how multiple neurons connect together with activation functions to make a robust neural network, we'll explore a teaching application known as the [TensorFlow Playground](https://playground.tensorflow.org/#activation=sigmoid&batchSize=10&dataset=gauss®Dataset=reg-plane&learningRate=0.03®ularizationRate=0&noise=0&networkShape=1&seed=0.10587&showTestData=false&discretize=true&percTrainData=50&x=true&y=true&xTimesY=false&xSquared=false&ySquared=false&cosX=false&sinX=false&c) [↗](https://playground.tensorflow.org/#activation=sigmoid&batchSize=10&dataset=gauss®Dataset=reg-plane&learningRate=0.03®ularizationRate=0&noise=0&networkShape=1&seed=0.10587&showTestData=false&discretize=true&percTrainData=50&x=true&y=true&xTimesY=false&xSquared=false&ySquared=false&cosX=false&sinX=false&c)
(<https://playground.tensorflow.org/#activation=sigmoid&batchSize=10&dataset=gauss®Dataset=reg-plane&learningRate=0.03®ularizationRate=0&noise=0&networkShape=1&seed=0.10587&showTestData=false&discretize=true&percTrainData=50&x=true&y=true&xTimesY=false&xSquared=false&ySquared=false&cosX=false&sinX=false&c>)

[osY=false&sinY=false&collectStats=false&problem=classification&initZero=false&hideText=false&discretize_hide=true®ularization_hide=true&learningRate_hide=true®ularizationRate_hide=true&percTrainData_hide=true&showTestData_hide=true&noise_hide=true&batchSize_hide=true](#)).

TensorFlow is a neural network and machine learning library for Python that has become an industry standard for developing robust neural network models. TensorFlow developed its playground application as a teaching tool to demystify the black box of neural networks and provide a working simulation of a neural network as it trains on a variety of different datasets and conditions.

After opening the link to the TensorFlow playground, does your screen match the following?



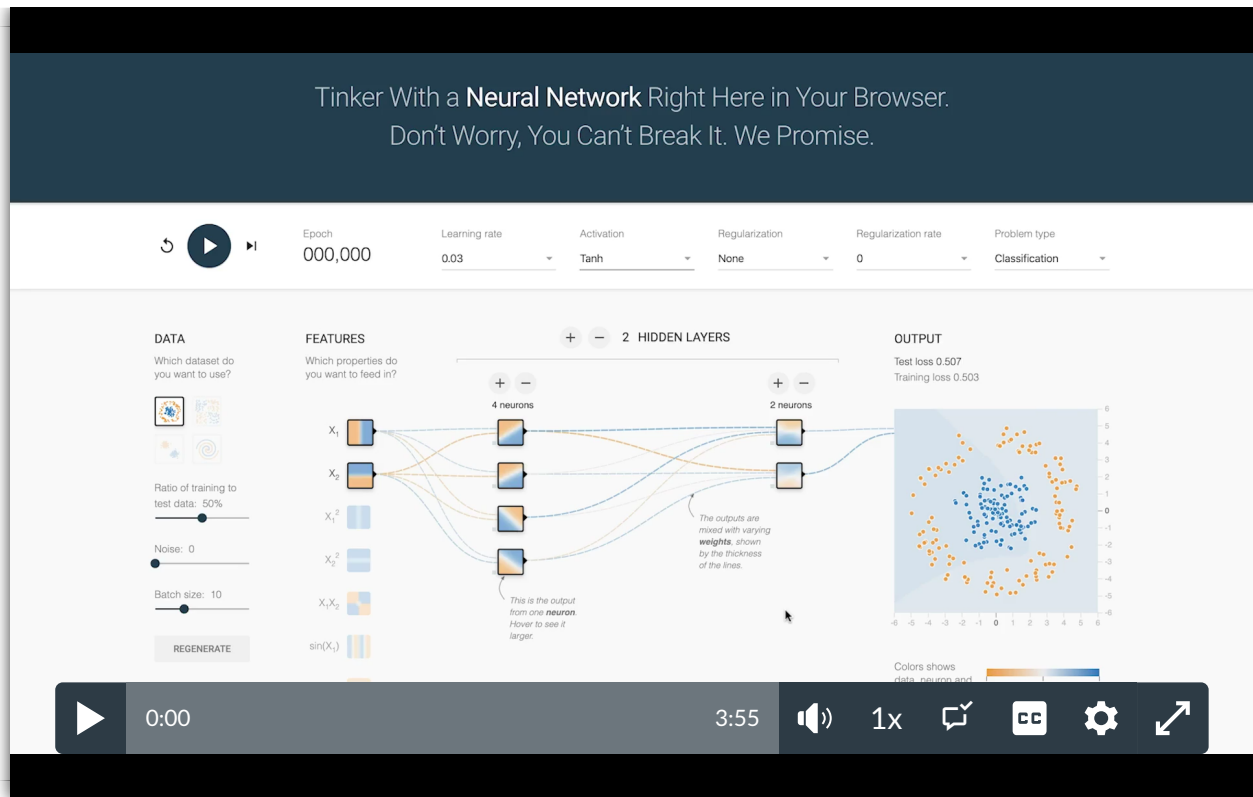
☐ Yes.

☐ No.

Check Answer

Finish ►

In this video, you'll use the simulations in the TensorFlow Playground to better understand how altering the neurons and activation functions of a neural network can change its performance.



Now that we have spent some time understanding the structure of a basic neural network and how each component impacts the final model, it is time to learn how to build our own functioning models. Don't worry if you feel like there are too many components, options, and parameters to keep track of—neural networks start off simple and grow to match the complexity of the input data.