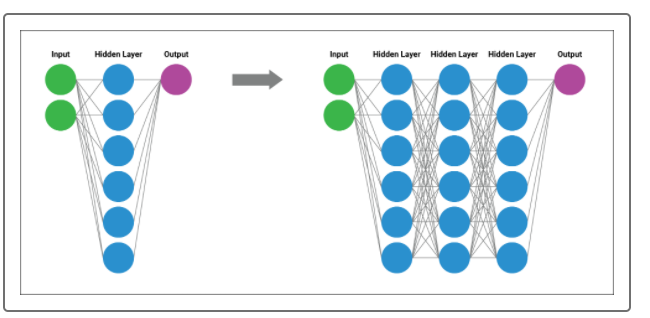
As we have seen, neural networks are designed to calculate weights for various input data and pass along an output value based on an activation function. With our basic neural networks, input data is parsed using an input layer, evaluated in a single hidden layer, then calculated in the output layer. In other words, a basic neural network is designed such that the input values are evaluated *only once* before they are used in a classification or regression equation. Although basic neural networks are relatively easy to conceptualize and understand, there are limitations to using a basic neural network, such as:

* A basic neural network with many neurons will require more training data than other comparable statistics and machine learning models to produce an adequate model.
* Basic neural networks struggle to interpret complex nonlinear numerical data, or data with many confounding factors that have hidden effects on more than one variable.
* Basic neural networks are incapable of analyzing image datasets without severe data preprocessing.

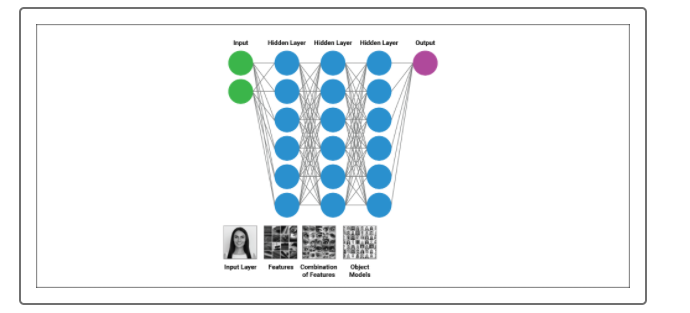
To address the limitations of the basic neural network, we can implement a more robust neural network model by adding additional hidden layers. A neural network with more than one hidden layer is known as a **deep neural network**:



Deep neural networks function similarly to the basic neural network, with one major exception. The outputs of one hidden layer of neurons (that have been evaluated and transformed using an activation function) become the inputs to additional hidden layers of neurons. As a result, the next layer of neurons can evaluate higher order interactions between weighted variables and identify complex, nonlinear relationships across the entire dataset. These additional layers can observe and weight interactions between clusters of neurons across the entire dataset, which means they can identify and account for more information than any number of neurons in a single hidden layer.

Deep neural network models also are commonly referred to as **deep learning models** due to their ability to learn from example data, regardless of the complexity or data input type. Just like humans, deep learning models can identify patterns, determine severity, and adapt to changing input data from a wide variety of data sources. Compared to basic neural network models, which require a large number of neurons to identify nonlinear characteristics, deep learning models only need a few neurons across a few hidden layers to identify the same nonlinear characteristics.

In addition, deep learning models can train on images, natural language data, soundwaves, and traditional tabular data (data that fits in a table or DataFrame), all with minimal preprocessing and direction:



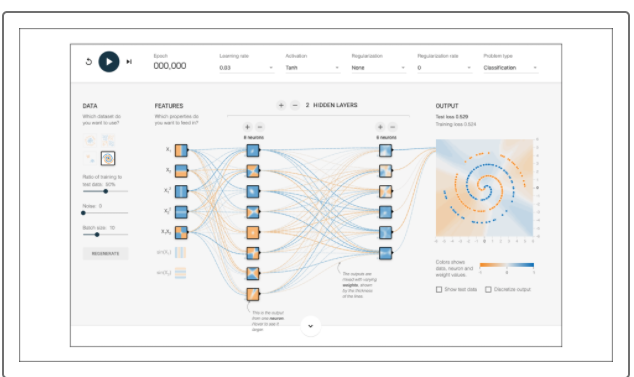
The best feature of deep learning models is its capacity to systematically process multivariate and abstract data while simultaneously achieving performance results that can mirror or even exceed human-level performance. It is no wonder why huge tech, pharmaceutical, and finance companies turn to deep learning models to evaluate and interpret datasets that were once unusable by traditional modelling techniques.

As with basic neural network models, deep learning models are not a new concept. However, deep learning models were not a feasible option for data scientists until implementation became easier with libraries like TensorFlow, and computing power became more affordable. Deep learning models require significantly longer training iterations and memory resources than their basic neural network counterparts, which allow for the deep learning models to achieve higher degrees of accuracy and precision. In other words, deep learning models may have more upfront costs, but they also have higher performance potential.

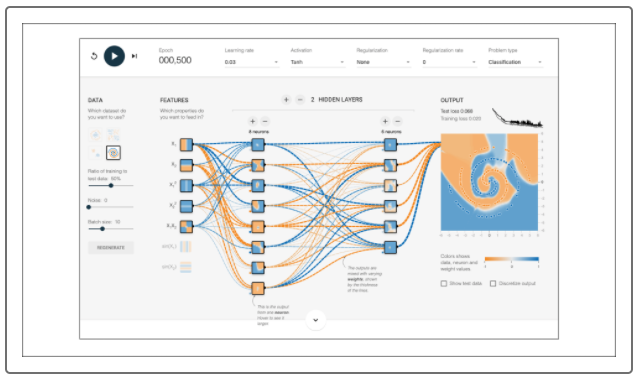
To conceptualize how performance differs between the basic neural network model versus a deep learning model, we'll return to the [TensorFlow Playground](https://playground.tensorflow.org/#activation=tanh&batchSize=10&dataset=spiral&regDataset=reg-plane&learningRate=0.03&regularizationRate=0&noise=0&networkShape=8&seed=0.14370&showTestData=false&discretize=false&percTrainData=50&x=true&y=true&xTimesY=true&xSquared=true&ySquared=true&cosX=false&sinX=false&cosY=false&sinY=false&collectStats=false&problem=classification&initZero=false&hideText=false), where our TensorFlow neural network model will try to classify a far more complex dataset such as a spiral. Since we're trying to build a binary classifier on nonlinear data, we'll use the tahn activation function rather than the sigmoid activation function. Additionally, we'll look at a couple of other mathematical inputs such as X1^2, X2^2, and X1X2, which will allow for our neural network model to train and identify patterns using nonlinear inputs. To start, let's allow our model to train over roughly 500 epochs:



Notice that by using a single hidden layer over 500 epochs, the model does a decent job building a classification model with loss around 0.2. In some cases this model would be sufficient, but when it comes to industry leaders such as Google and Apple, model performance must be near perfection. Therefore, we must try and build a more robust model by designing a deep learning model with two layers. In your TensorFlow Playground, add an additional hidden layer with six neurons—these will analyze the outputs of our eight neurons in the first layer to try and boost performance:



Now that we have our updated deep learning model, let's try to train the neural network over the same 500 epochs:



Notice that our deep learning model was able to reduce the loss from roughly 0.2 to 0.07, which could mean the difference of 80% classification accuracy to 95% and above! Let's try adding an additional layer with four more neurons and train the new model over 500 epochs.



Looking at the results of our simulated deep learning model, it does not appear that adding more layers increased the overall performance of the model. This is because the additional layer was redundant—the complexity of the dataset was encapsulated within the two hidden layers. Adding layers does not always guarantee better model performance, and depending on the complexity of the input data, adding more hidden layers will only increase the chance of overfitting the training data. Unfortunately, there is no easy solution or rule of thumb to identify how many layers are required to maximize performance. The only way to determine how "deep" the deep learning model should be is through trial and error. You must train and evaluate a model with increasingly deeper and deeper layers until the model no longer demonstrates noticeable improvements over the same number of epochs.