TensorFlow Playground: Neural Network Exploration

Devin Prikryl-Martin

Houston Community College

 ITAI 1378 Computer Vision

Professor Patricia McManus

October 3, 2024

TensorFlow Playground: Neural Network Exploration

Neural networks are a type of AI system developed in order for computers to mimic the way neurons work in a human brain. A neural network is comprised of a number of components that allow it to receive an input, process that input and generate an output. The most basic part of the network is a neuron, or a “node”, which are grouped and organized into multiple layers. There is always an input layer and an output layer with a varying number of hidden layers in between them. Each node is connected to every node in the subsequent layer and those connections all have a numerical value or “weight” that signifies the strength of the connection between two nodes. Each node is also acted on by an Activation Function which decides mathematically whether or not the data is valuable and should be passed to the next layer. (Sharma, 2022) During the training of a neural network, information is input and travels forward through the layers to the output, which is then compared to the expected result. The difference between the expected result and the output is calculated as something called a loss function. The algorithm then works backwards through the layers from output to input, calculating how much each weight contributed to the error and adjusting the weights to produce better predictions. This process repeats a number of times during training and is then tested with a reserve set of data to see how well the network outputs a correct prediction on data it wasn’t trained on. (3Blue1Brown, 2017)

The first task involves creating a simple neural network with a single hidden layer and experimenting with the different activation functions. Activation functions are mathematical functions that make decisions about what information should be passed to the next layer. There are four activation functions to choose from inside of TensorFlow Playground. We begin by reducing the number of hidden layers from two to one, leaving all of the other settings at default, since our goal for this task is just to examine the effect of the activation functions. The default activation function is Tanh. (Jain, 2019)

A screenshot of a computer

Description automatically generated

Image 1 above shows the result of running the neural network with the Tanh activation function. In about 2,601 Epochs the output is returning an extremely low test loss at .003, which shows that the network is classifying the data pretty accurately and this is reflected in the data visualized in orange and blue. Next we run the Sigmoid activation function.

A screenshot of a computer

Description automatically generated

Image 2 shows the result of running the sigmoid activation function, which after 3,356 Epochs showed the output test loss at .005, where it stalled. I was trying to see how many more iterations it would take this function to yield the same results as the Tanh experiment. Again, we see that the network is functioning well, having properly classified the orange and blue data. Now we will try the ReLU function.

A screenshot of a computer

Description automatically generated

Image 3 shows that the ReLU function returns a much faster result, yielding the .003 test loss output at 774 Epochs, which suggests the network is able to classify data much more quickly when using this function. Interestingly, the visual data shows us a much more geometrical segmentation of the data than the previous functions. Finally we will try the linear function.

A screenshot of a computer

Description automatically generated

Image 4 shows the linear activation function. After running for over 3,500 Epochs it yields exactly the same output as it did in the beginning, showing no change to the test loss calculation or the visual data. This is to represent how ineffective a linear function is at this task.

We then begin the second task, which involves manipulating both the number of neurons in the hidden layers, as well as the number of hidden layers. Based on what I have already learned I would theorize that adding neurons and layers increases both the speed and accuracy of the output. I will begin by returning the activation function to the default Tanh setting and adding neurons to the single hidden layer. The model will allow a maximum of 8 neurons in each layer so that is the number I will try, which I will compare to my results from image 1.

A screenshot of a computer

Description automatically generated

Image 5 shows us the results of adding 4 neurons to the single hidden layer. In less than half of the Epochs as our initial run in image 1, we are able to achieve the same test loss output. This supports my theory that increasing the neurons increases the speed. Next I will add a hidden layer. I will return the first layer to its original 4 neuron state and add another hidden layer with 4 neurons so that I can make appropriate comparisons.

A screenshot of a computer

Description automatically generated

Image 6 shows a much faster output at the .003 test loss reading that I am using as the base for comparison, at just less than 300 epochs. That is nearly 10x fewer Epochs, which further supports my theory that increasing neurons and layers means that the networks reach accurate predictions with significantly less training. Next I will max out the potential hidden layers to see how quickly the model yields results.

A screenshot of a computer

Description automatically generated

Image 7 shows that an addition of 4 more hidden layers produces an amazing .001 test loss output with only 110 Epochs.

For the third task I will be adjusting the learning rate and observing its impact on convergence speed and accuracy. Before I make any adjustments to the model I take some time to better understand what the purpose of a learning rate is, and to define the convergence speed and accuracy in this context. I find that a learning rate dictates the speed at which a neural network learns to solve a problem optimally. The larger a learning rate is, the faster it learns. But if it is too large it can lead to a dip in precision because it is learning so quickly it misses information. The smaller a learning rate is, the slower and more careful the process is. But if it is too small it causes extremely slow learning. Optimizing the learning rate involves experimenting with that component until it yields efficient training and high accuracy. The convergence speed is the rate at which a model improves its performance, often measured by how many epochs it takes to reach a certain level of accuracy. This is how I have been assessing the performance of the models thus far, just without knowing the technical term. The learning rate is presently set at 0.03, which is the mid-point option in the dropdown list. I will adjust the model so that there is only one hidden layer and 4 neurons in that layer so we can compare to our basic run in image 1. I then adjust the learning rate to its lowest rate and observe the results. (Statquest, 2017)

A screenshot of a computer

Description automatically generated

We can see from image 8 that slowing the learning rate down to the lowest rate causes the model to fail. The test loss is almost the same at 2700 Epochs as it was before running the model. Next I will increase the learning rate to its highest rate and observe.

A screenshot of a computer

Description automatically generated

In image 9 we can observe that increasing the learning rate to its max actually has an adverse effect on the convergence speed, increasing the test loss instead of reducing it. This signifies to me that the training is happening so quickly that it is not learning accurately.

The fourth task entails adjusting the data noise and taking note of how it affects the network’s ability to generalize. Data noise is extra information in the data that can make it harder for the model to make accurate predictions. For example, lowering the resolution of an image can produce noise because it obscures the edges of shapes, and neural networks often use edge detection to help them figure out what is contained in the image. In TensorFlow playground, the noise is represented by mixing a few blue dots in with the orange dots and vice versa. I hypothesize that this will confuse the model and make it more difficult to classify the data neatly. I begin by returning to default settings and increasing the noise to 10. (Wright, 2024)

A screenshot of a computer

Description automatically generated

In image 10 we can see based on the visual representation of the data and the test loss of .058 that even adding a little bit of noise makes it much more difficult for the model to classify the data accurately. Next I will increase the noise to 20 and observe the results.

A screenshot of a computer

Description automatically generated

In image 11 we can observe that the test loss is getting worse as the noise increases. Next I will max out the noise and take note of the results.

A screenshot of a computer

Description automatically generated

In image 12 we can see the effect that maxing out the noise has on the model. It significantly reduces the accuracy of the test loss output.

In the final task I will explore the effect using a different data set has on the model output under the same basic parameters I have established. I will run all three alternative data sets and then compare them to each and the initial run in image 1.

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated

In images 13-15 we can see the visual representation of how well the different data was classified by this extremely simple neural network. The more clearly the data is separated before the model runs has a huge impact on the speed and accuracy of the model’s output. In image 14, for example, we can see that the clearly separated data produces a test loss of .000 at only 163 Epochs. Compared to the more interwoven data in image 15 which produces only a .1 change in test loss at the ~2,500 Epoch mark I’ve been using as a base. This shows us how important it is to the speed and accuracy of the model for the data to be preprepared.

This assignment has taught me a lot about what sorts of components a neural network is made of, what sorts of adjustments can be made to those components and what sorts of effects those adjustments can have on the function, speed and accuracy of a model. It is clear that a model must have appropriate parameters for the data it is given to produce a valuable result. The greatest challenges I faced during this assignment was breaking down some of the more technical concepts into lay terms that I could understand. I relied heavily on youtube videos to try to explain these concepts to me in multiple ways so I would have the best chance at understanding them. Overall I feel I am walking away from this assignment with a much deeper understanding of how a neural network functions.

References

3Blue1Brown. (2017, October 5). But what is a neural network? | Chapter 1, Deep learning [Video]. YouTube. <https://www.youtube.com/watch?v=aircAruvnKk>

Jain, V. (2019, December 30). *Everything you need to know about “activation functions” in Deep learning models*. Medium. https://towardsdatascience.com/everything-you-need-to-know-about-activation-functions-in-deep-learning-models-84ba9f82c253

Sharma, S. (2022, November 20). *Activation functions in neural networks*. Medium. https://towardsdatascience.com/activation-functions-neural-networks-1cbd9f8d91d6

Statquest. (2017, September 12). Learning Rate in a Neural Network explained [Video]. YouTube. <https://www.youtube.com/watch?v=jWT-AX9677k&t=36s>

Wright, G. (2024, April 12). *What is noisy data?: Definition from TechTarget*. Business Analytics. https://www.techtarget.com/searchbusinessanalytics/definition/noisy-data#:~:text=Noisy%20data%20unnecessarily%20increases%20the,data%20and%20facilitate%20data%20mining.

Anthropic. (2024, October 3). Claude. “Explain how a neural network functions in lay terms”