TensorFlow Playground Presentation

Part 1 Introduction:

Neural Networks computational models that were inspired by the human brain's neural form. They are essential in the field of machine learning, because they have an ability to learn difficult patterns and make decisions based on data. Some key components of neural networks are neurons, layers, activation functions, etc. Neurons are the basic unit, they receive all the input signals, process them using weights, which are the strengths of connections, and at the end create an output signal. Each neuron often has a linear transformation followed by a non-linear activation function that establishes nonlinearity into the network. Then there are the layers, which have neural networks organized, and each layer contains many neurons. The input layer receives information to pass to the hidden layer, which then performs complex computations, and lastly the output layer produces the output results. Activation functions are functions that decide neuron outputs, they have non-linearities that enable them to model difficult relationships in data like sigmoid, which helps binary classification tasks with values between 0 and 1. Rectified linear unit helps output inputs straight if it is positive, if not then it is zero, and Tanh is similar to sigmoid, but with the values of -1 and 1. Lastly, there are weights and biases, which help during the training process to minimize error of predictions and results, plus allowing adjustments of the output independently of the input data. All these components are significant, because each have a role to play in order for the neural networks to process and learn from data, they perform complex tasks that other algorithms have a hard time handling in many domains.

Part 2 Tasks & Observations**:**

- **Task 1** - For this task, we were to gain exposure to neural networks and to experiment with multiple activation functions.
    - **Observation** - We evaluated Sigmoid[1], TanH[2], and ReLU[3] activation functions using our single hidden layer network with four neurons. We observed that in each iteration, the neurons passed a different distribution of weights to the output; with Sigmoid passing 3/4 negative, TanH passing 2/4 negative, and ReLU passing all 4 weights as negative.
- **Task 2** - Here we experimented with making adjustments to the number of neurons in our hidden layer, as well as increasing the number of hidden layers in our model.
    - **Observation** - To make an even comparison, we remained with ReLU as our activation function but we scaled out our hidden layers and neurons to four each. As a result, we observed lower test loss and training loss values when compared with our smaller model using the same dataset and activation function. This suggests that larger models containing additional neurons & layers could provide higher degrees of precision.[4]

- **Task 3** - Our challenge with this task was to adjust the learning rate and to witness the effects of the adjustments on our model, with respect to speed and accuracy.
    - **Observation** - We tweaked the previous model's learning rate by an order of magnitude, from 0.03 to 0.003. We hypothesized that a 10x reduction in learning rate would result in a proportional increase in the time taken to achieve the same Test & Training loss values; and we were proven correct in our assumption.[5]
- **Task 4** - Data noise in the context of neural networks refers to random variations, irrelevant information, or errors that are present in the training data.
    - **Observation** - We reverted our learning rate from the previous task to .03 and increased the noise value from 0 to 20, and this was the first time we observed our model seem to really struggle. We observed our model's Test loss value held constant at 0.082 beginning at 747 epoch, suggesting it had reached convergence, and therefore would not be able to reduce the error rate further.
- **Task 5** - Finally, we evaluated additional different datasets with our model.
    - **Observation** - Our findings were largely identical when comparing the linear datasets, however with the spiral dataset we saw a dramatic increase in our loss curve. In an effort to solve for this and achieve an acceptable loss value, we began to experiment with additional hyperparameters as well as the distribution of neurons within each hidden layer. Finally, we found our solution was two-fold:
        - Implemented a decreasing number of neurons from left to right
        - Implemented L1 regularization
    - With these adjustments, we were able to achieve an acceptable loss value[7]

Part 3: Parameter Changes

Neural Networks performance is deeply affected by many factors including activation functions, neuron counts, learning rates data noise, and datasets. Activation functions for example, affect how well a network can apprehend non-linear relationships in data. Rectified Linear Unit is a popular activation function, because of its efficiency in gradient computation and prevents vanishing gradients, leading to faster training and improved performance. Problems can arise when vanishing gradients occurs, because it can lead to slow learning, or another issue would be choosing an inappropriate activation function for a particular task can cause overly sparse activations, impacting negatively both training and performance. Neuron counts and layer depths determine the complexity and capacity to learn complicated patterns. Increasing them can improve the abilities to capture intricate relationships and nuances within the input, enhancing performance on sophisticated feature extractions. Though if too much is implemented it can lead to longer training sessions and become impractical for deployment in applications. Learning rates are hyper-parameters used to determine the pace at which the algorithm will learn values of a parameter approximation. It commands how fast a network can adjust its weights based on gradient descent. It is very important to choose a good learning rate for example, choosing a too small learning rate can result in a slow learning process, while a big one can mean a loss value does not converge, leading to problems in the learning process. Choosing the right one can better the efficiency and effectiveness of training networks. Data noise are random variations in inout data that are capable of concealing basic patterns that can

reduce model generalization. It challenges network robustness, which requires techniques like dropouts to reduce overloads. It enhances generalization and calibration by making it more resilient to minor variations and outliers in the input, and also forces the networks to learn more invariant features. Lastly, dataset size is the amount of labeled data that is available for training, that influences directly the ability to learn patterns and generalize. It can be beneficial by providing diverse examples for the model to learn from and larger ones help lessen overfitting. Some issues that can occur still are storage problems with bigger datasets or overfitting with smaller ones. Balancing all these factors is essential in order for neural networks to succeed in their ability to perform complex tasks on applications and ensure robustness in real life deployments.

Part 4: Implications

Parameter changes in neural networks have various components like learning rate, batch size, number of layers, activation functions, and more. These components have important implications in real-world scenarios. For example in tasks like image classification or natural language processing the learning rate can affect how quickly or slow a model converges to a specific solution, meaning higher learning rates can speed up convergence but can risk overshooting, while lower ones can ensure a stabled convergence but require more training. Neural networks are useful in these applications, because it gathers insights, the meanings from text data and documents, and breaks image parts to look for patterns. Another real life example is in applications like autonomous driving systems the use of neural networks can work for object detection and decision-making. If you increase the number of layers or neurons, it can enhance the complexity of the model and feature extraction techniques. This allows for more accurate predictions, but will require careful tuning in case of overfitting. Still, it is still very useful and has changed the way transportation is handled today by controlling things like steering, braking, and accelerating. It also helps identify trees, traffic lights, and other things using patterns and the data that includes images from the cameras on the self-driving cars. Last example is where tasks involve different data size, like financial forecasting models that are based on time-series data, this would require batch size to impact memory utilization and training efficiency. When larger batch size are applied it can expedite training but can sacrifice generalization meanwhile, smaller batches offer more accurate gradients but with slower convergence. Neural networks help aid in stock prediction, fraud detection, and credit scoring. It can analyze past stock market information to predict future trends, making it useful for investors and businesses. These are just some of the many examples on how parameters can be applied to real life situations, improving performance, efficiency, and adaptability. Parameter changes are important, because they allow neural networks to be assigned to specific tasks and maximizes their use in various fields from healthcare, finance, automotive, etc.

Conclusion:

In conclusion, we learned that neural networks excel at being able to recognize patterns in complex information, making them great for image and speech recognition, natural language processing, etc. They have the adaptability to learn form big chunks of information without being

programmed to unlike other algorithms, and are very flexible in various applications, due to them being able to model highly nonlinear relationships. Their behavior throughout the experiment was quite interesting because the more you added different types of layers or approached it with different datasets it would defer the weights creation until the shape of the input is known, and be able to fine-tune it better for optimal performance. Adding more layers can expand the capacity to learn complex structures and relationships, and is very flexible with different data formats. Some challenges we faced were how to track the patterns shown and see how everything worked, because it is a new concept that can get confusing if you are not familiar with it. We overcame it with trying out different datasets and other methods to see how different the results would be, and it helped us understand a little bit more. We also researched a bit more about neural networks and how they work to have the necessary knowledge to understand what is happening.

References:

Brownlee, Jason. “Understand the Impact of Learning Rate on Neural Network Performance.” *MachineLearningMastery.Com*, 11 Sept. 2020, machinelearningmastery.com/understand-the-dynamics-of-learning-rate-on-deep-learning-neural-networks/.

“What Is a Neural Network?” IBM, 6 Oct. 2021, www.ibm.com/topics/neural-networks.

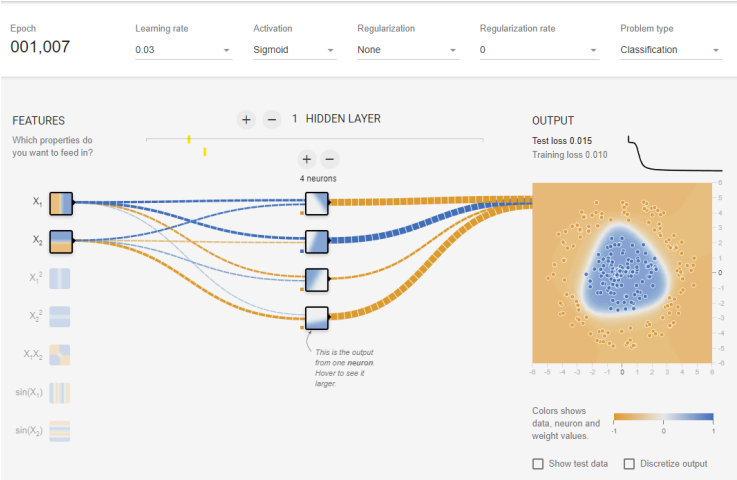
Wright, Gavin. “What Is Noisy Data?: Definition from TechTarget.” *Business Analytics*, TechTarget, 12 Apr. 2024, www.techtarget.com/searchbusinessanalytics/definition/noisy-data.

.“." Mathematics. . Encyclopedia.Com. 15 Jun. 2024 .” *Encyclopedia.Com*, Encyclopedia.com, 1 July 2024, www.encyclopedia.com/education/news-wires-white-papers-and-books/graphs-and-effects-parameter-changes.

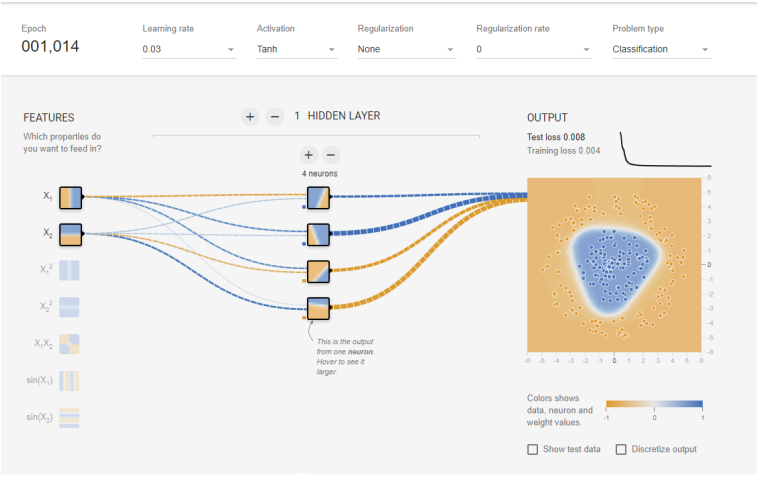
“What Is Image Classification? Basics You Need to Know.” *SuperAnnotate*, www.superannotate.com/blog/image-classification-basics. Accessed 1 July 2024.

Appendix

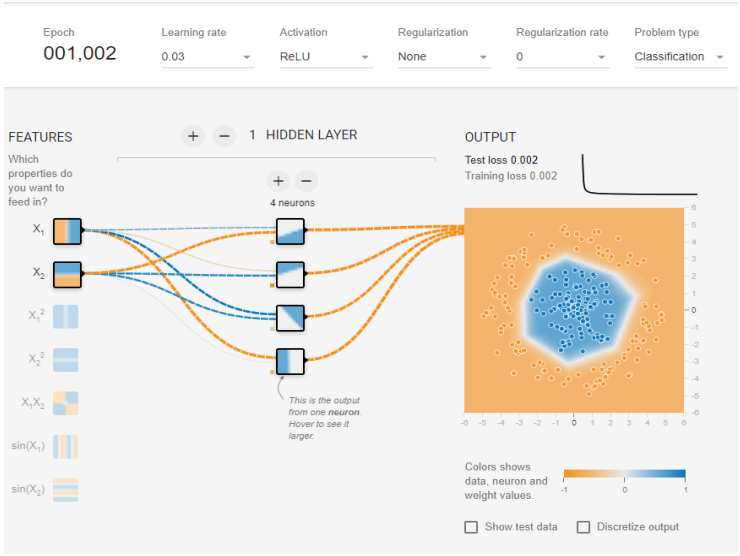
[1] Task 1, Sigmoid Activation



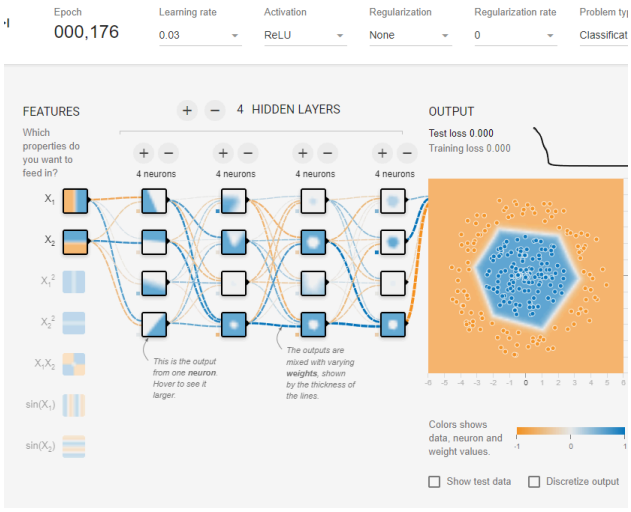
[2] Task 1, Tanh Activation



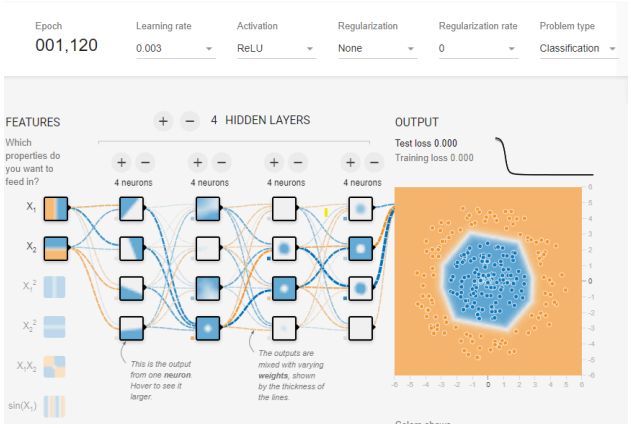
[3] Task 1, ReLU Activation



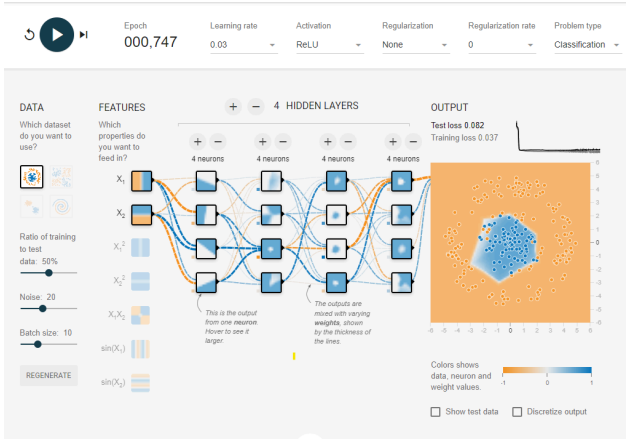
[4] Task 2, Increasing Neuron & Layer count



[5] Task 3, Modifying Learning Rate



[6] Task 4, Introducing Noisy Data



[7] Task 5, Spiral Dataset