Andre Ellis

Sha’Rise Giggs

Carlos Granillo

Olowatoyin Abolarin

Mike Huynh

Enrique Quintero

Computer Vision Artificial Intel- ITAI 1378

Prof. Patricia McManus

10/02/2024

|  |  |
| --- | --- |
| TensorFlow playground | Abstract  The objective of this assignment is to familiarize you with the fundamentals of neural networks and their behavior.  Group: Great Achievers |

1. **Brief Introduction to Neural Networks and Their Components**

Neural networks are the synthetic essence of machine learning and foundational learning ideas in the context of artificial intelligence. They are computer models that are inspired by the structure and functions of the human brain. Such networks are built to learn from the data, recognize complex patterns, and solve complicated problems. To explain the mechanisms further, the neuron is the basic unit where the input data are processed and output received. The connection strength among the neurons is signified by weights, which are subject to modification.

Layers refer to a neural network viewed in layers, where there are mainly three types to be considered.

Input Layer: This is the injection point of data into the neuron.

Hidden Layers: These perform transformations and computations on the data. One or more hidden layers of neurons may exist to get along in learning the alteration of complexity in presentation.

Output Layer: The Output Layer gives the final output of the network, generally regression, classification, and so on.

Activation Function: This function gives information regarding whether a neuron will be activated or not after the computation of the weighted sum of all inputs. This can also help the network in learning complex patterns.

Networks have enormous importance in many areas, including language, vision, and healthcare. Progressing history in such neural networks is a demonstration of guilt-free performance and accuracy toward solving the problems especially defined against conventional algorithms.

1. **Description of each task, along with screenshots or visualizations**

***Linear:*** Used for output in regression problems, where a continuous value is predicted.

Gráfico, Gráfico de cajas y bigotes

Descripción generada automáticamente

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

***Sigmoid***: Ideal for the last layer of binary classification.

Diagrama

Descripción generada automáticamente

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

***Tanh***: Useful in hidden layers to normalize data (when activations should be between -1 and 1).

Gráfico, Gráfico de cajas y bigotes

Descripción generada automáticamente

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

***ReLU***: Widely used in hidden layers of deep networks for its speed and ability to mitigate the vanishing gradient problem.

Diagrama

Descripción generada automáticamente

* Which to use and when?

Sigmoid: Ideal for the last layer of binary classification. (0 or 1)

Tanh: Useful in hidden layers to normalize data (when activations should be between -1 and 1).

ReLU: Widely used in hidden layers of deep networks for its speed and ability to mitigate the vanishing gradient problem. (Range: 0 a ∞).

Linear: Used for output in regression problems, where a continuous value is predicted. Range (∞ a ∞)

1. **Detailed Observations and explanations for the effects of parameter changes**

* Activation Functions:

We experimented with different activation functions but two are more focused for the observations: ReLU and Sigmoid. ReLU allows only positive numbers to pass through, converting negative numbers to zero. Using ReLU, the network learned quickly and achieved higher accuracy. In contrast, Sigmoid converts numbers to values between 0 and 1. When we used Sigmoid, the network learned much slower and sometimes even stopped learning. We also learned that ReLU was better for the neural network because it facilitated faster learning, especially in larger models.

* Number of Neurons:

When we changed the number of neurons in the hidden layer and used fewer neurons, the network struggled to learn enough from the data and did not perform well, a situation known as underfitting. However, adding more neurons allowed the network to capture complex patterns, leading to improved accuracy. We discovered that while more neurons can enhance performance, having too many can result in overfitting, where the model excels on training data but fails on new data. Therefore, finding a balance is essential.

* Learning Rate:

We also adjusted the learning rate, which determines how quickly the model updates during training. A high learning rate sped up training but often led to instability, as the model skipped over the best solutions. Conversely, a low learning rate provided steady learning and better accuracy but took longer. This taught our group that the learning rate is crucial; a high rate can destabilize training, while a low rate ensures consistent progress.

* Data Noise:

To examine the effect of data noise, we adjusted the "Noise" slider. With low noise, the network achieved high accuracy because the data was clear. However, increasing noise reduced accuracy since it became challenging for the model to learn the true patterns. We eventually realized that excessive noise could hinder the network's learning ability, making it difficult to generalize and predict accurately.

* Different Datasets:

Finally, we tested various datasets, including linear, circle, and spiral datasets. The linear dataset was straightforward, allowing the network to learn quickly without many neurons. The circle dataset was more complex, requiring additional neurons and careful tuning. The spiral dataset was the most challenging, demanding a significant number of neurons and layers to produce good results due to its complicated patterns. We were comprehended that the dataset type directly influences the difficulty level for the network; simpler datasets require fewer resources, while more complex datasets necessitate larger, deeper networks.

1. **Discussion of practical implications**

TensorFlow Playground gives ones the opportunity to investigate the various parameters of neural networks and their various real-world applications. Even relatively small changes in parameters such as activation functions, learning rates, numbers of hidden layers and neurons, data noise, and dataset choice can have a tremendous effect on neural network efficiency and ability in various contexts.

The most important implication is the significance of activation functions. For instance, ReLU may be employed to speed up learning in image classification problems, a common task in computer vision applications. Knowledge about the best fitting activation function for a problem can help develop a better generalizing model, such as face recognition or object detection. In doing so, the activation function will be altered accordingly to optimize performance for other tasks as well. Hidden layers and neurons strike a delicate balance between capacity and overfitting. This is practically applicable in natural language processing (NLP); thus, architecture selection allows the model to learn complicated patterns in text input instead of memorizing them. With this background, one realizes that models can be neither too simple nor too complex, which appeals especially to problems such as sentiment analysis and machine translation.

Next, the selection of good learning rates is another contender with significant implications. In many production situations, a well-tuned learning rate can help converge the model in the quickest time without sacrificing accuracy; insights gained from the experience of just how learning rates can affect performance will be invaluable in assisting all developers in fine-tuning their approach.The assessment of data noise gives us an important insight. Real-life data is never clean, and knowing how noise affects a model's capacity to meet its generalization requirements is of tremendous utility. For instance, in autonomous driving systems, where sensor data may contain noise from the outside world, training models that coexist with those perturbations enhance dependability and safety in real usage.

Besides, the comprehensive lessons learned on the importance of dataset selection are worth mentioning. Because of the shape of the dataset being evaluated, not all data is created equal. As a consequence, the quality of the training and testing datasets may impact the ability of the neural network quite considerably.

To conclude, the experience collected with neural network parameters is of great importance to practical applications. This is because this knowledge does not simply enhance the architecture of neural networks but also contributes to the advancement of AI technologies across various sectors. Thus, the present and the future are very important when it comes to further improvement."

1. **Conclusion & Challenges**

In conclusion, this hands-on experience with neural networks in TensorFlow Playground enabled our group to gain deeper awareness into how certain parameters such as learning rate and number of neurons could hugely affect model performance. The group was able to note that complexity, in terms of additional layers on the model, would probably lead to a case of overfitting whereby the model would perform well on training data but generally poorly on unseen test data. Experience provides an excellent opportunity to retain firmly in our minds the importance of balancing complication with simplicity for good generalization. In trying out different settings, we have gained such hands-on knowledge of how neural networks learn and change.

Speaking of the ongoing challenges, in TensorFlow Playground, there is fundamentally an understanding of how various parameters affect the performance of a model. Such parameters include the learning rate, the number of neurons per layer, and the number of layers. Particularly to a novice, understanding how these components engage and how they affect the outcome is confusing. A high learning rate may speed up training while risking missing the best solution, while a low learning rate leads to long convergence. While such manipulability is beneficial, it requires a certain level of experience and experimentation to grasp adequately where each parameter has its main effects.

Another issue that arises is overfitting-almost always associated with an overly complicated model, likely due to very high numbers of layers or neurons. Thus, though the model exhibits very good fitting to the training set, it cannot generalize properly to unseen data. This might be really frustrating for beginners as they might expect better performance only to find their model is performing poorly on the test set. Recognizing signs of overfitting and knowing when to simplify a model are very important lessons that take time to learn.

It is sometimes unfortunate that the convergence of a model cannot be understood in practice because, even when learning rate and architecture are appropriate, it could still happen that the model will not converge to any reasonable solution. In general, this is possible for two reasons: too high a learning rate, thus skipping over the optimal parameters, or a very small learning rate, which would make it converge quite slowly. An expert must patiently strike a balance among all these aspects and continuously keep testing to actually make the model work.

1. **References**

Explained: Neural networks. (2017, April 14). MIT News | Massachusetts Institute of Technology. <https://news.mit.edu/2017/explained-neural-networks-deep-learning-0414>

*Deep Learning*. (n.d.). https://www.deeplearningbook.org/